UMSIDA PRESS

BUKU AJAR

Kecerdasan Buatan/Artificial Intelegent (AI)

PENULIS

Hindarto
Mochamad Alfan Rosid

Buku Ajar
Kecerdasan Buatan/Artificial Intelegent (AI)

Oleh

Hindarto

Sumarno

Mochamad Alfan Rosid

Diterbitkan oleh
UMSIDA PRESS
Jl. Mojopahit 666 B Sidoarjo
ISBN: 978-623-464-034-2

Buku Ajar
Kecerdasan Buatan/Artificial Intelegent (AI)

Penulis :
Hindarto
Sumarno
Mochamad Alfan Rosid

ISBN :
978-623-464-034-2

Editor :
M.Tanzil Multazam,S.H.,M.Kn
Mahadika Darmawan,KW,.S.Pd,.M.Pd

Copy Editor :
Wiwit Wahyu Wijayanti

Design Sampul dan Tata Letak :
Wiwit Wahyu Wijayanti

Penerbit :
UMSIDA Press

Redaksi :
Universitas Muhammadiyah Sidoarjo
Jl. Mojopahit No 666B Sidoarjo, Jawa Timur
Cetakan pertama, Agustus 2022

PRAKATA

Buku ajar ini ditulis sebagai buku panduan dan referensi mahasiswa Informatika untuk mengambil mata kuliah Kecerdasan Buatan. Kelebihan buku ajar ini yaitu terdapat soal-soal yang ada pada bagian bab di buku ajar ini dan terdapat contoh-contoh aplikasi yang berhubungan dengan mata kuliah kecerdasan buatan. Sasaran utama buku ajar ini adalah mahasiswa Informatika dan sasaran umum buku ajar ini adalah para dosen dan praktisi yang ingin belajar tentang Kecerdasan Buatan. Prasyarat agar dapat menggunakan buku ajar ini tidak ada. Penulisan buku ajar Kecerdasan Buatan/Artificial Intelegent (AI) ini ditulis dalam 9 BAB yang berisi:

Bab 1 Pengenalan Kecerdasan Buatan (AI)

Dalam bab 1 ini dijelaskan tentang Konsep Dasar / Pengertian AI, Asumsi Dasar AI, Perbedaan antara Pemrograman Konvensional dengan AI, Bidang-bidang Aplikasi AI, Representasi masalah, Karakteristik masalah, Sistem produksi, dan Konsep *State Space.*

Bab 2 Masalah Dan Metode Pemecahan Masalah

Dalam bab 2 dijelaskan tentang Represanti masalah, Representasi pengurangan masalah, dan mendefinisikan masalah dalam suatu ruang keadaan

Bab 3 Strategi Pencarian atau Penelusuran (*searching*)

Dalam bab 3 ini dijelaskan tentang macam-macam Metode pencarian yang ada di AI, diantaranya : B*lind search* (Pencarian Buta), *Depth First Search*, *Breadth First Search*, *Nondeterministic Search*, *Heuristic Search*, *Beam Search*, *Hill Climbing*, dan *Best First Search*

Bab 4 Representasi Pengetahuan

Dalam bab 4 ini dijelaskan tentang Pengetahuan Prosedural vs Deklaratif, *Logic Programming*, *Production Rules*, dan *Forward maupun Backward Reasoning Matching*

Bab 5 Teori Ketidakpastian

Dalam bab 5 ini dijelaskan tentang Teorema Bayes, Certainty Factor, Teori Dempster Shafer.

Bab 6 Sistem Pakar

Dalam bab 6 ini dijelaskan tentang Konsep Dasar Sistem Pakar, Komponen Utama Sistem Pakar, dan Bidang-bidang aplikasi Sistem Pakar

Bab 7 Logika fuzzy

Dalam bab 7 ini dijelaskan tentang Himpunan Fuzzy Operator Fuzzy, Fuzzy Inference System, Fuzzy Inference System metode Tsukamoto.

Bab 8 Jaringan Syaraf Tiruan (JST)

Dalam bab 8 ini dijelaskan tentang Konsep Dasar JST, Komponen utama JST, Bidang-bidang aplikasi JST

Bab 9 Algoritma Genetika

Dalam bab 8 ini dijelaskan tentang Pengertian Algoritma Genetika, susunan neuron yang ada pada Algoritma Genetika, fungsi aktivasi pada Algoritma Genetika, arsitektur jaringan pada Algoritma Genetika, aplikasi – apliaksi pada Algoritma Genetika.

Buku ajar ini dibaca dari awal bab sampai akhir, sehingga antar bab ada yang saling berkaitan. Buku ajar ada buku pendamping yang saling berhubungan, diantaranya buku Sistem Pakar, Logika Fuzzy, Jaringan Suaraf tiruan, algoritam genetika.

Dengan selesainya penulisan buku ajar ini penulis mengucapkan terima kasih kepada semua pihak yang telah memberikan bahan-bahan tulisan baik langsung maupun tidak langsung. Penulis juga mengucapkan terima kasih khususnya kepada:

1. Dr. Hidayatullah, M.Si pemangku pimpinan tertinggi yaitu Rektor Universitas Muhammadiyah Sidoarjo yang telah memberikan dan memfasilitasi dalam penulisan buku ajar ini.
2. LP3iK Universitas Muhammadiyah Sidoarjo yang telah memfasilitasi dan mengkoordinasi dalam penulisan buku ajar ini.
3. Dr. Hindarto, S.Kom, MT. sebagai Dekan Fakultas Sains dan Teknologi, Universitas Universitas Muhammadiyah Sidoarjo yang telah memberikan dukungan untuk mengikuti penulisan buku ajar ini.
4. Ir. Sumarno, MM. sebagai Kepala Program Studi Informatika, Universitas Universitas Muhammadiyah Sidoarjo yang telah memberikan dukungan untuk mengikuti penulisan buku ajar ini.

Akhir kata, kritik dan saran sangat diharapkan untuk penyempurnaan buku ajar ini. Harapan kami semoga buku ajar ini dapat digunakan sebagai tambahan informasi dan bermanfaat bagi aktivitas pembelajaran mata kuliah Kecerdasan Buatan di Program Studi Informatika, Fakultas Sains dan Teknologi, Universitas Muhammadiyah Sidoarjo.

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

Bab 1
PengenalanKecerdasan Butan

1.1. Pendahuluan

Umat manusia telah menamai dirinya sendiri dengan nama ilmiah homo sapiens, manusia yang bijak, karena kapasitas mental kita sangat penting bagi kehidupan sehari-hari dan perasaan diri kita. Bidang kecerdasan buatan atau Artificial Intelegent (AI), mencoba memahami entitas cerdas. Jadi, salah satu alasan untuk mempelajarinya adalah untuk belajar lebih banyak tentang diri kita sendiri. Namun berbeda dengan filosofi dan psikologi yang juga mementingkan kecerdasan, AI berupaya membangun entitas cerdas sekaligus memahaminya. Alasan lain untuk mempelajari AI adalah bahwa entitas cerdas yang dibangun ini menarik dan berguna dengan sendirinya. AI telah menghasilkan banyak produk yang signifikan dan mengesankan bahkan pada tahap awal perkembangannya. Meskipun tidak ada yang bisa memprediksi masa depan secara rinci, jelas bahwa komputer dengan kecerdasan setingkat manusia (atau lebih baik) akan berdampak besar pada kehidupan kita sehari-hari dan pada masa depan peradaban. AI menangani salah satu teka-teki pamungkas. Bagaimana mungkin otak yang lambat dan kecil, baik biologis maupun elektronik, memahami, memprediksi, dan memanipulasi dunia yang jauh lebih besar dan lebih rumit dari pada dirinya sendiri? Bagaimana cara kita membuat sesuatu dengan property tersebut ? Ini adalah pertanyaan sulit, tetapi tidak seperti pencarian untuk perjalanan yang lebih cepat dari cahaya atau perangkat anti gravitasi, peneliti di AI memiliki bukti kuat bahwa pencarian itu mungkin dilakukan. Yang harus dilakukan peneliti adalah bercermin untuk melihat contoh system cerdas.

AI adalah salah satu disiplin ilmu terbaru. Ini secara resmi dimulai pada tahun 1956, ketika nama itu diciptakan, meskipun pada saat itu pekerjaan telah dilakukan selama sekitar lima tahun. Seiring dengan genetika modern, ini secara teratur dikutip sebagai "bidang yang paling diinginkan" oleh para ilmuwan di disiplin lain. Seorang siswa di bidang fisika mungkin secara masuk akal merasa bahwa semua ide bagus telah diambil oleh Galileo, Newton, Einstein, dan yang lainnya, dan membutuhkan waktu bertahun-tahun untuk belajar sebelum seseorang dapat menyumbangkan ide-ide baru. AI, di sisi lain, masih memiliki celah untuk Einstein penuh waktu.

Studi tentang kecerdasan juga merupakan salah satu disiplin ilmu tertua. Selama lebih dari 2000 tahun, filosofi telah mencoba memahami bagaimana melihat, belajar, mengingat, dan bernalar dapat, atau harus, dilakukan. ' Munculnya komputer yang dapat digunakan di awal 1950-an mengubah spekulasi yang terpelajar tetapi bersandar tentang kemampuan mental ini menjadi disiplin teoritis dan eksperimental yang nyata. Banyak yang merasa bahwa "Otak Super Elektronik" yang baru memiliki potensi kecerdasan yang tidak terbatas. "Faster dan Einstein" adalah tajuk utama yang umum. Tapi selain menyediakan kendaraan untuk menciptakan entitas kecerdasan buatan, komputer menyediakan alat untuk menguji teori kecerdasan, dan banyak teori gagal bertahan dalam pengujian kasus "keluar dari kursi, kedalam api". AI ternyata lebih sulit daripada yang dibayangkan pada awalnya, dan sebagai hasilnya, ide-ide modem jauh lebih kaya, lebih halus, dan lebih menarik.

AI saat ini mencakup berbagai macam sub bidang, dari area tujuan umum seperti persepsi dan penalaran logis, hinggat ugas-tugas khusus seperti bermain catur, membuktikan teorema matematika, menulispuisi, dan mendiagnosis penyakit. Sering kali, para ilmuwan di bidang lain secara bertahap beralih kekecerdasan buatan, di mana mereka menemukan alat dan kosakata untuk mensistematisasikan dan mengotomatiskan tugas-tugas intelektual yang telah mereka kerjakan sepanjang hidup mereka. Demikian pula, pekerja di AI dapat memilih untuk menerapkan metode mereka kebidang upaya intelektual manusia mana pun. Dalam pengertian ini, AI benar-benar bidang universal.

Gambar 1.1 Wilayah Kerja dari Kecerdasan Buatan

1.2 Sejarah Kecerdasan Buatan (AI)

Tahun 1666, Matematikawan dan filsuf Gottfried Leibniz menerbitkan Dissertatio de arte combinatoria (Tentang Seni Kombinasi), mengikuti Ramon Llull dalam mengusulkan alfabet pemikiran manusia dan menyatakan bahwa semua ide tidak lain adalah kombinasi dari sejumlah kecil konsep sederhana.

Pada tahun 1726, Jonathan Swift menerbitkan Gulliver's Travels, yang mencakup deskripsi Engine, sebuah mesin di pulau Laputa (dan parodi dari ide-ide Llull): "sebuah Proyek untuk meningkatkan Pengetahuan spekulatif dengan Operasi praktis dan mekanis." Dengan menggunakan "Temuan" ini, "Orang yang paling bodoh dengan biaya yang masuk akal, dan dengan sedikit kerja fisik, dapat menulis Buku dalam Filsafat, Puisi, Politik, Hukum, Matematika, dan Teologi, dengan bantuan paling sedikit dari Jenius atau studi. "

Tahun 1763, Thomas Bayes mengembangkan kerangka berpikir tentang kemungkinan kejadian. Inferensi Bayesian akan menjadi pendekatan terdepan dalam pembelajaran mesin.

Tahun 1854, George Boole berpendapat bahwa penalaran logis dapat dilakukan secara sistematis dengan cara yang sama seperti memecahkan sistem persamaan.

Tahun 1898, pameran listrik di Madison Square Garden yang baru saja selesai, Nikola Tesla membuat demonstrasi kapal pertama di dunia yang dikendalikan radio. Perahu itu dilengkapi dengan, seperti yang digambarkan Tesla, "pikiran yang dipinjam."

Tahun 1914, Insinyur Spanyol Leonardo Torres y Quevedo mendemonstrasikan mesin catur pertama, yang mampu menjadi raja dan benteng melawan raja akhir permainan tanpa campur tangan manusia.

1921 Penulis Ceko Karel apek memperkenalkan kata "robot" dalam dramanya R.U.R. (Robot Universal Rossum). Kata “robot” berasal dari kata “robota” (bekerja).

Tahun 1925, Houdina Radio Control merilis mobil tanpa pengemudi yang dikendalikan radio, berjalan-jalan di New York City.

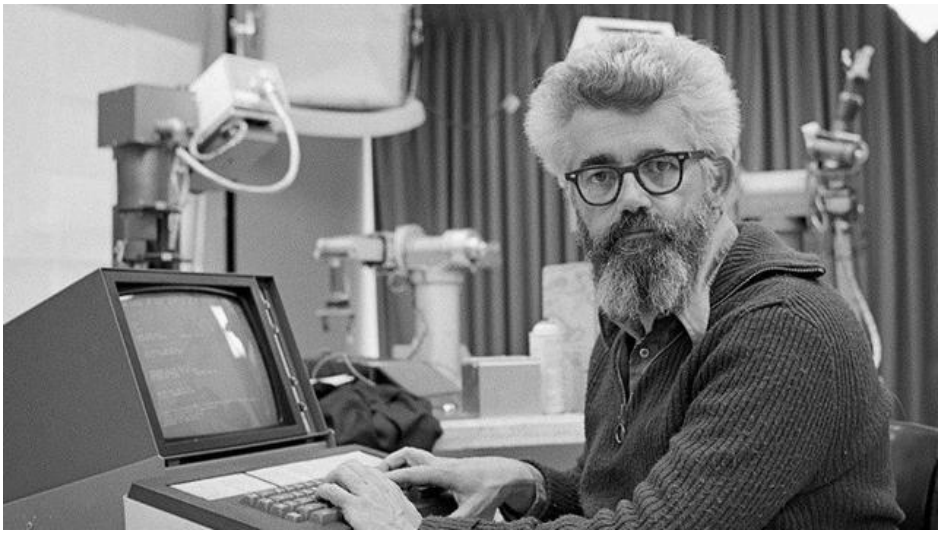
Tahun 1927, Film fiksi ilmiah Metropolis dirilis. Ini menampilkan robot ganda dari seorang gadis petani, Maria, yang melepaskan kekacauan di Berlin tahun 2026 — itu adalah robot pertama yang digambarkan dalam film, menginspirasi tampilan Art Deco dari C-3PO di Star Wars.

Tahun 1929, Makoto Nishimura mendesain Gakutensoku, bahasa Jepang untuk "belajar dari hukum alam", robot pertama yang dibuat di Jepang. Itu bisa mengubah ekspresi wajahnya dan menggerakkan kepala dan tangannya melalui mekanisme tekanan udara.

Tahun 1943, Warren S. McCulloch dan Walter Pitts menerbitkan "A Logical Calculus of the Ideas Immanent in Nervous Activity" dalam Buletin Biofisika Matematika. Makalah berpengaruh ini, di mana mereka membahas jaringan "neuron" buatan yang diidealkan dan disederhanakan dan bagaimana mereka dapat melakukan fungsi logis sederhana, akan menjadi inspirasi untuk "jaringan saraf" berbasis komputer (dan kemudian "pembelajaran mendalam") dan deskripsi populer mereka sebagai "meniru otak."

Tahun 1949, Edmund Berkeley menerbitkan Giant Brains: Or Machines That Think di mana dia menulis: “Baru-baru ini ada banyak berita tentang mesin raksasa aneh yang dapat menangani informasi dengan kecepatan dan keterampilan yang luar biasa….Mesin ini mirip dengan apa yang akan dilakukan oleh otak. jika itu terbuat dari perangkat keras dan kawat, bukan daging dan saraf… Sebuah mesin dapat menangani informasi; itu dapat menghitung, menyimpulkan, dan memilih; itu dapat melakukan operasi yang wajar dengan informasi. Oleh karena itu, sebuah mesin dapat berpikir.”

Tahun 1949, Donald Hebb menerbitkan Organisasi Perilaku: Teori Neuropsikologis di mana ia mengusulkan teori tentang pembelajaran berdasarkan dugaan mengenai jaringan saraf dan kemampuan sinapsis untuk memperkuat atau melemahkan dari waktu ke waktu.



Gambar 1.2 Lahirnya Ilmuwan Komputer

Tahun 1950, "Pemrograman Komputer untuk Bermain Catur" karya Claude Shannon adalah artikel pertama yang diterbitkan tentang pengembangan program komputer permainan catur.
Tahun 1950, Alan Turing menerbitkan "Mesin Komputasi dan Kecerdasan" di mana ia mengusulkan "permainan tiruan" yang kemudian dikenal sebagai "Tes Turing."

Tahun 1951, Marvin Minsky dan Dean Edmunds membangun SNARC (Stochastic Neural Analog Reinforcement Calculator), jaringan saraf tiruan pertama, menggunakan 3000 tabung vakum untuk mensimulasikan jaringan 40 neuron.

Tahun 1952, Arthur Samuel mengembangkan program permainan catur komputer pertama dan program komputer pertama yang belajar sendiri.

31 Agustus 1955 Istilah "kecerdasan buatan" diciptakan dalam proposal untuk "2 bulan, studi 10 orang tentang kecerdasan buatan" yang diajukan oleh John McCarthy (Dartmouth College), Marvin Minsky (Universitas Harvard), Nathaniel Rochester (IBM), dan Claude Shannon (Laboratorium Telepon Bell). Lokakarya, yang berlangsung setahun kemudian, pada Juli dan Agustus 1956, umumnya dianggap sebagai tanggal lahir resmi lapangan baru.

Desember 1955 Herbert Simon dan Allen Newell mengembangkan Logic Theorist, program kecerdasan buatan pertama, yang akhirnya akan membuktikan 38 dari 52 teorema pertama di Whitehead dan Russel's Principia Mathematica.

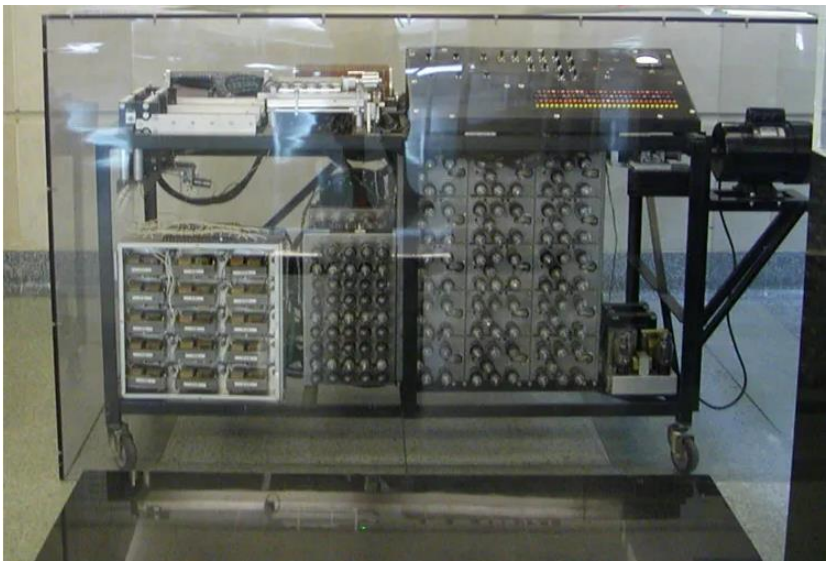
Tahun 1957, Frank Rosenblatt mengembangkan Perceptron, jaringan saraf tiruan awal yang memungkinkan pengenalan pola berdasarkan jaringan pembelajaran komputer dua lapis. The New York Times melaporkan Perceptron menjadi "embrio komputer elektronik yang [Angkatan Laut] harapkan akan dapat berjalan, berbicara, melihat, menulis, mereproduksi dirinya sendiri dan menyadari keberadaannya." The New Yorker menyebutnya sebagai "mesin yang luar biasa ... mampu melakukan apa yang perlu dipikirkan."

Tahun 1958, John McCarthy mengembangkan bahasa pemrograman Lisp yang menjadi bahasa pemrograman paling populer yang digunakan dalam penelitian kecerdasan buatan.

Tahun 1959, Arthur Samuel menciptakan istilah "pembelajaran mesin," yang melaporkan pemrograman komputer "sehingga ia akan belajar memainkan permainan catur yang lebih baik daripada yang dapat dimainkan oleh orang yang menulis program itu."

Tahun 1959, Oliver Selfridge menerbitkan "Pandemonium: Sebuah paradigma untuk pembelajaran" dalam Prosiding Simposium Mekanisasi Proses Pemikiran, di mana ia menggambarkan model untuk proses dimana komputer dapat mengenali pola yang belum ditentukan sebelumnya.

Tahun 1959 John McCarthy menerbitkan “Programs with Common Sense” dalam Proceedings of the Symposium on Mechanization of Thought Processes, di mana ia menggambarkan Advice Taker, sebuah program untuk memecahkan masalah dengan memanipulasi kalimat dalam bahasa formal dengan tujuan akhir membuat program “yang belajar dari pengalaman mereka seefektif manusia.”



Gambar 1.3 Komputer raksasa dai Insinyur Jerman

Tahun 1961, Robot industri pertama, Unimate, mulai bekerja di jalur perakitan di pabrik General Motors di New Jersey. Tahun 1961, James Slagle mengembangkan SAINT (Symbolic Automatic INTegrator), sebuah program heuristik yang memecahkan masalah integrasi simbolik dalam kalkulus mahasiswa baru.

Tahun 1964, Daniel Bobrow menyelesaikan disertasi PhD MIT-nya yang berjudul “Input Bahasa Alami untuk Sistem Pemecahan Masalah Komputer” dan mengembangkan STUDENT, sebuah program komputer pemahaman bahasa alami.

Tahun 1965, Herbert Simon meramalkan bahwa "mesin akan mampu, dalam waktu dua puluh tahun, melakukan pekerjaan apa pun yang dapat dilakukan manusia."

Tahun 1965, Hubert Dreyfus menerbitkan "Alkimia dan AI," dengan alasan bahwa pikiran tidak seperti komputer dan bahwa ada batas di mana AI tidak akan berkembang.

Tahun 1965 I.J, Good menulis dalam "Speculations Concerning the First Ultraintelligent Machine" bahwa "mesin ultraintelligent pertama adalah penemuan terakhir yang perlu dibuat manusia, asalkan mesin itu cukup jinak untuk memberi tahu kita cara mengendalikannya."

Tahun 1965, Joseph Weizenbaum mengembangkan ELIZA, sebuah program interaktif yang melakukan dialog dalam bahasa Inggris tentang topik apa pun. Weizenbaum, yang ingin mendemonstrasikan kedangkalan komunikasi antara manusia dan mesin, terkejut dengan banyaknya orang yang menghubungkan perasaan seperti manusia dengan program komputer.

Tahun 1965, Edward Feigenbaum, Bruce G. Buchanan, Joshua Lederberg, dan Carl Djerassi mulai mengerjakan DENDRAL di Universitas Stanford. Sistem pakar pertama, mengotomatiskan proses pengambilan keputusan dan perilaku pemecahan masalah ahli kimia organik, dengan tujuan umum mempelajari pembentukan hipotesis dan membangun model induksi empiris dalam sains.

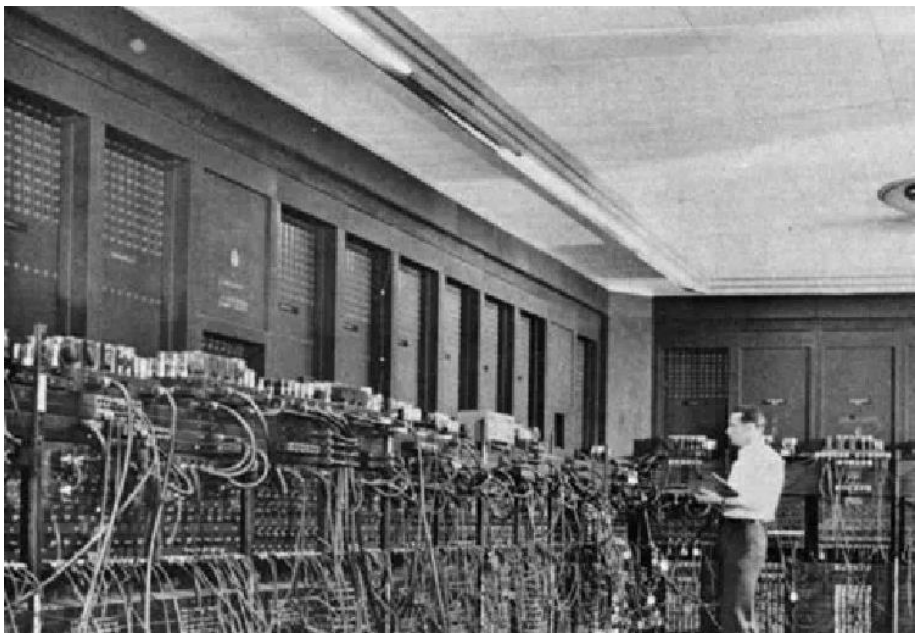
Tahun 1966, Shakey si robot adalah robot bergerak serba guna pertama yang mampu menjelaskan tindakannya sendiri. Dalam sebuah artikel majalah Life tahun 1970 tentang "manusia elektronik pertama" ini, Marvin Minsky dikutip mengatakan dengan "kepastian": "Dalam tiga hingga delapan tahun kita akan memiliki mesin dengan kecerdasan umum rata-rata manusia."

Tahun 1968, Film 2001: Space Odyssey dirilis, menampilkan Hal, sebuah komputer hidup.

Tahun 1968, Terry Winograd mengembangkan SHRDLU, program komputer pemahaman bahasa alami awal.

Tahun 1969, Arthur Bryson dan Yu-Chi Ho menggambarkan backpropagation sebagai metode optimasi sistem dinamis multi-tahap. Sebuah algoritma pembelajaran untuk jaringan saraf tiruan multi-layer, telah memberikan kontribusi yang signifikan terhadap keberhasilan pembelajaran mendalam di tahun 2000-an dan 2010-an, setelah daya komputasi cukup maju untuk mengakomodasi pelatihan jaringan besar.

Tahun 1969, Marvin Minsky dan Seymour Papert menerbitkan Perceptrons: An Introduction to Computational Geometry, menyoroti keterbatasan jaringan saraf sederhana. Dalam edisi yang diperluas yang diterbitkan pada tahun 1988, mereka menanggapi klaim bahwa kesimpulan 1969 mereka secara signifikan mengurangi pendanaan untuk penelitian jaringan saraf: “Versi kami adalah bahwa kemajuan telah terhenti secara virtual karena kurangnya teori dasar yang memadai… Tahun 1960-an ada banyak sekali eksperimen dengan perceptron, tetapi tidak ada yang bisa menjelaskan mengapa mereka bisa mengenali jenis pola tertentu dan bukan yang lain.”



Gambar 1.4 Komputer Eniac (Elektronic Numerical Integrator and Calculator)

Tahun 1970, Robot antropomorfik pertama, WABOT-1, dibangun di Universitas Waseda di Jepang. Ini terdiri dari sistem kontrol anggota badan, sistem penglihatan dan sistem percakapan.
Tahun 1972, MYCIN, sistem pakar awal untuk mengidentifikasi bakteri penyebab infeksi parah dan merekomendasikan antibiotik, dikembangkan di Universitas Stanford.
Tahun 1973, James Lighthill melaporkan kepada British Science Research Council tentang penelitian kecerdasan buatan negara bagian, menyimpulkan bahwa "tidak ada bagian dari bidang ini yang sejauh ini menemukan penemuan yang menghasilkan dampak besar yang kemudian dijanjikan," yang menyebabkan berkurangnya dukungan pemerintah secara drastis untuk penelitian AI .
Tahun 1976, Ilmuwan komputer Raj Reddy menerbitkan “Speech Recognition by Machine: A Review” dalam Proceedings of the IEEE, meringkas karya awal Natural Language Processing (NLP).
Tahun 1978, Program XCON (eXpert CONfigurer), sistem pakar berbasis aturan yang membantu pemesanan komputer VAX DEC dengan memilih komponen secara otomatis berdasarkan kebutuhan pelanggan, dikembangkan di Carnegie Mellon University.
Tahun 1979, Stanford Cart berhasil melintasi ruangan yang dipenuhi kursi tanpa campur tangan manusia dalam waktu sekitar lima jam, menjadi salah satu contoh paling awal dari kendaraan otonom.

Gambar 1.5 Komputer Pribadi tahun 1970 an

Tahun 1980, Wabot-2 dibangun di Universitas Waseda di Jepang, robot humanoid musisi yang mampu berkomunikasi dengan seseorang, membaca skor musik dan memainkan nada dengan kesulitan rata-rata pada organ elektronik.

Tahun 1981, Kementerian Perdagangan dan Industri Internasional Jepang menganggarkan $850 juta untuk proyek Komputer Generasi Kelima. Proyek ini bertujuan untuk mengembangkan komputer yang dapat melakukan percakapan, menerjemahkan bahasa, menafsirkan gambar, dan nalar seperti manusia.

Tahun 1984, Electric Dreams dirilis, sebuah film tentang cinta segitiga antara seorang pria, seorang wanita dan komputer pribadi.

Tahun 1984, Pada pertemuan tahunan AAAI, Roger Schank dan Marvin Minsky memperingatkan akan datangnya “AI Musim Dingin”, meramalkan ledakan gelembung AI (yang memang terjadi tiga tahun kemudian), mirip dengan pengurangan investasi AI dan dana penelitian di pertengahan 1970-an.

Tahun 1986, Mobil tanpa pengemudi pertama, sebuah van Mercedes-Benz yang dilengkapi dengan kamera dan sensor, dibangun di Universitas Bundeswehr di Munich di bawah arahan Ernst Dickmanns, melaju hingga 55 mph di jalan-jalan kosong.

Oktober 1986 David Rumelhart, Geoffrey Hinton, dan Ronald Williams menerbitkan "Representasi pembelajaran dengan kesalahan propagasi mundur," di mana mereka menggambarkan "prosedur pembelajaran baru, propagasi balik, untuk jaringan unit mirip neuron."

Tahun 1987, Video Knowledge Navigator, yang menyertai pidato utama CEO Apple John Sculley di Educom, membayangkan masa depan di mana "aplikasi pengetahuan akan diakses oleh agen pintar yang bekerja melalui jaringan yang terhubung ke sejumlah besar informasi digital."

Tahun 1988, Judea Pearl menerbitkan Penalaran Probabilistik dalam Sistem Cerdas. Kutipan Penghargaan Turing 2011-nya berbunyi: “Judea Pearl menciptakan fondasi representasional dan komputasional untuk pemrosesan informasi di bawah ketidakpastian. Dia dikreditkan dengan penemuan jaringan Bayesian, formalisme matematika untuk mendefinisikan model probabilitas yang kompleks, serta algoritma utama yang digunakan untuk inferensi dalam model ini. Pekerjaan ini tidak hanya merevolusi bidang kecerdasan buatan tetapi juga menjadi alat penting bagi banyak cabang teknik dan ilmu alam lainnya.”

Tahun 1988, Rollo Carpenter mengembangkan Jabberwacky bot obrolan untuk "mensimulasikan obrolan manusia alami dengan cara yang menarik, menghibur, dan lucu." Ini adalah upaya awal untuk menciptakan kecerdasan buatan melalui interaksi manusia.

Tahun 1988, Anggota IBM T.J. Watson Research Center menerbitkan “Pendekatan statistik untuk terjemahan bahasa,” yang menandai pergeseran dari metode terjemahan mesin berbasis aturan ke probabilistik, dan mencerminkan pergeseran yang lebih luas ke “pembelajaran mesin” berdasarkan analisis statistik dari contoh-contoh yang diketahui, bukan pemahaman dan “pemahaman ” dari tugas yang ada (proyek IBM Candide, yang berhasil menerjemahkan antara bahasa Inggris dan Prancis, didasarkan pada 2,2 juta pasang kalimat, sebagian besar dari proses bilingual parlemen Kanada).

Tahun 1988, Marvin Minsky dan Seymour Papert menerbitkan edisi yang diperluas dari buku mereka Perceptrons tahun 1969. Dalam “Prologue: A View from 1988” mereka menulis: “Salah satu alasan mengapa kemajuan begitu lambat di bidang ini adalah karena para peneliti yang tidak terbiasa dengan sejarahnya terus membuat banyak kesalahan yang sama seperti yang dilakukan orang lain sebelumnya.”

Tahun 1989, Yann LeCun dan peneliti lain di AT&T Bell Labs berhasil menerapkan algoritma backpropagation ke jaringan saraf multi-layer, mengenali kode ZIP tulisan tangan. Mengingat keterbatasan perangkat keras pada saat itu, dibutuhkan waktu sekitar 3 hari (masih merupakan peningkatan yang signifikan dari upaya sebelumnya) untuk melatih jaringan.

Gambar 1.6 Komputer Pribadi tahun 1980 an

Tahun 1990, Rodney Brooks menerbitkan “Elephants Don't Play Chess,” mengusulkan pendekatan baru untuk AI—membangun sistem cerdas, khususnya robot, dari bawah ke atas dan berdasarkan interaksi fisik berkelanjutan dengan lingkungan: “Dunia adalah yang terbaik untuk dirinya sendiri. model… Triknya adalah merasakannya dengan tepat dan cukup sering.”

Tahun 1993, Vernor Vinge menerbitkan “The Coming Technological Singularity,” di mana ia memprediksi bahwa “dalam tiga puluh tahun, kita akan memiliki sarana teknologi untuk menciptakan kecerdasan manusia super. Tak lama setelah itu, era manusia akan berakhir. ”

Tahun 1995, Richard Wallace mengembangkan chatbot A.L.I.C.E (Entitas Komputer Internet Linguistik Buatan), terinspirasi oleh program ELIZA Joseph Weizenbaum, tetapi dengan penambahan pengumpulan data sampel bahasa alami pada skala yang belum pernah terjadi sebelumnya, dimungkinkan oleh munculnya Web.

Tahun 1997, Sepp Hochreiter dan Jürgen Schmidhuber mengusulkan Long Short-Term Memory (LSTM), jenis jaringan saraf berulang yang digunakan saat ini dalam pengenalan tulisan tangan dan pengenalan suara.

Tahun 1997, Deep Blue menjadi program permainan catur komputer pertama yang mengalahkan juara catur dunia.

Tahun 1998, Dave Hampton dan Caleb Chung menciptakan Furby, robot domestik atau hewan peliharaan pertama.

Tahun 1998, Yann LeCun, Yoshua Bengio dan lain-lain menerbitkan makalah tentang penerapan jaringan saraf untuk pengenalan tulisan tangan dan mengoptimalkan backpropagation.

Gambar 1.7 A.L.I.C.E (Artificial Linguistic Internet Computer Entity)

Tahun 2000, MIT Cynthia Breazeal mengembangkan Kismet, sebuah robot yang dapat mengenali dan mensimulasikan emosi.

Tahun 2000, Robot ASIMO Honda, robot humanoid dengan kecerdasan buatan, mampu berjalan secepat manusia, mengantarkan nampan ke pelanggan di restoran.

Tahun 2001, AI Artificial Intelligence dirilis, sebuah film Steven Spielberg tentang David, android seperti anak kecil yang diprogram secara unik dengan kemampuan untuk mencintai.

Tahun 2004, DARPA Grand Challenge pertama, sebuah kompetisi hadiah untuk kendaraan otonom, diadakan di Gurun Mojave. Tak satu pun dari kendaraan otonom menyelesaikan rute 150 mil.

Tahun 2006, Oren Etzioni, Michele Banko, dan Michael Cafarella menciptakan istilah "pembacaan mesin," mendefinisikannya sebagai "pemahaman teks otonom" yang secara inheren tidak diawasi.

Tahun 2006, Geoffrey Hinton menerbitkan “Learning Multiple Layers of Representation,” merangkum ide-ide yang mengarah ke “jaringan saraf multilayer yang berisi koneksi top-down dan

melatih mereka untuk menghasilkan data sensorik daripada mengklasifikasikannya," yaitu, pendekatan baru untuk mendalam sedang belajar.

Tahun 2007, Fei Fei Li dan rekan-rekannya di Universitas Princeton mulai merakit ImageNet, database besar gambar beranotasi yang dirancang untuk membantu penelitian perangkat lunak pengenalan objek visual.

Tahun 2009, Rajat Raina, Anand Madhavan dan Andrew Ng menerbitkan "Pembelajaran Tanpa Pengawasan Dalam Skala Besar menggunakan Prosesor Grafis," dengan alasan bahwa "prosesor grafis modern jauh melampaui kemampuan komputasi CPU multicore, dan memiliki potensi untuk merevolusi penerapan metode pembelajaran tanpa pengawasan yang mendalam ."

Tahun 2009, Google mulai mengembangkan, secara rahasia, mobil tanpa pengemudi. Pada tahun 2014, itu menjadi yang pertama lulus, di Nevada, tes mengemudi sendiri negara bagian AS.

Tahun 2009, Ilmuwan komputer di Intelligent Information Laboratory di Northwestern University mengembangkan Stats Monkey, sebuah program yang menulis berita olahraga tanpa campur tangan manusia.

Gambar 1.8 ASIMO (Advanced Step in Innovative Mobility) is a humanoid robot created by Honda in 2000

Tahun 2010, Peluncuran ImageNet Large Scale Visual Recognition Challenge (ILSVCR), kompetisi pengenalan objek AI tahunan.

Tahun 2011, Jaringan saraf convolutional memenangkan kompetisi Pengenalan Tanda Lalu Lintas Jerman dengan akurasi 99,46% (vs manusia pada 99,22%).

Tahun 2011 Watson, komputer penjawab pertanyaan bahasa alami, bersaing di Jeopardy! dan mengalahkan dua mantan juara.

Tahun 2011, Para peneliti di IDSIA di Swiss melaporkan tingkat kesalahan 0,27% dalam pengenalan tulisan tangan menggunakan jaringan saraf convolutional, peningkatan yang signifikan atas tingkat kesalahan 0,35% -0,40% di tahun-tahun sebelumnya.

Juni 2012 Jeff Dean dan Andrew Ng melaporkan percobaan di mana mereka menunjukkan jaringan saraf yang sangat besar 10 juta gambar tidak berlabel yang diambil secara acak dari video YouTube, dan "untuk hiburan kami, salah satu neuron buatan kami belajar untuk merespons gambar dengan kuat.. . kucing."

Oktober 2012 Jaringan saraf convolutional yang dirancang oleh para peneliti di University of Toronto mencapai tingkat kesalahan hanya 16% dalam Tantangan Pengenalan Visual Skala Besar ImageNet, peningkatan yang signifikan atas tingkat kesalahan 25% yang dicapai oleh entri terbaik tahun sebelumnya.

Maret 2016 AlphaGo Google DeepMind mengalahkan juara Go Lee Sedol.

Gambar 1.9 Eugene si Turing yang mengalahkan 'komputer manusia' – dengan kata-kata 'miliknya' sendiri

1.3 Perbedaan antara Pemrograman Konvensional dengan Kecerdasan Buatan (AI)

Pemrograman komputer tradisional telah ada selama lebih dari satu abad, dengan program komputer pertama yang diketahui berasal dari pertengahan tahun 1800-an. Pemrograman Tradisional mengacu pada program yang dibuat secara manual yang menggunakan data masukan dan berjalan di komputer untuk menghasilkan keluaran.

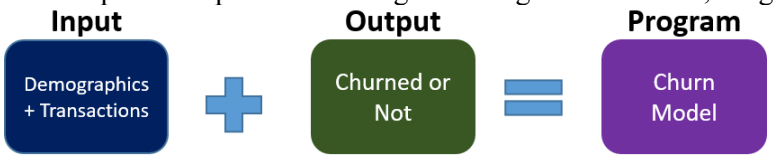
Tetapi selama beberapa dekade sekarang, jenis pemrograman tingkat lanjut telah merevolusi bisnis, terutama di bidang kecerdasan dan analitik embedded. Dalam Machine Learning, juga dikenal sebagai analytics tambahan, data input dan output dimasukkan ke algoritma untuk membuat program. Ini menghasilkan wawasan yang kuat yang dapat digunakan untuk memprediksi hasil di masa depan.

Pemrograman tradisional adalah proses manual artinya seseorang (programmer) menciptakan program. Tetapi tanpa siapa pun yang memprogram logika, seseorang harus merumuskan atau membuat kode aturan secara manual. Memiliki data masukan, dan seseorang (pemrogram) membuat kode program yang menggunakan data tersebut dan berjalan di komputer untuk menghasilkan keluaran yang diinginkan.

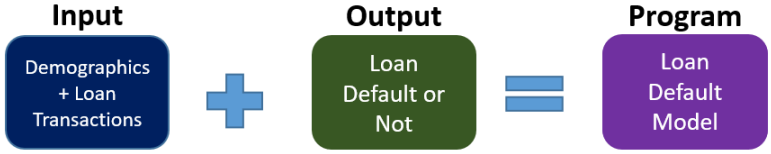
Pembelajaran AI, di sisi lain, data input dan output diumpankan ke algoritma untuk membuat program.

inilah perbedaan mendasar antara pemrograman tradisional dan pembelajaran AI. Tanpa ada yang memprogram logika, Dalam pemrograman Tradisional seseorang harus secara manual merumuskan / kode aturan sementara dalam Pembelajaran AI algoritma secara otomatis merumuskan aturan dari data, yang sangat kuat.

Misalnya, jika Anda memasukkan demografi pelanggan, transaksi sebagai input dan output yang diamati jika mereka berputar atau tidak di masa lalu, algoritma akan merumuskan program yang akan mengetahui cara memprediksi apakah seseorang akan bergolak atau tidak, Program itu disebut model.



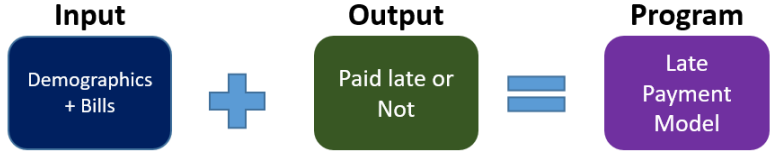
Pelanggan, secara umum, memiliki banyak data input dan output dan mereka dapat memasukkannya ke algoritma untuk membuat model prediktif. Misalnya, feed-in informasi pelanggan / transaksi pinjaman (input) dan berapa banyak yang gagal pada pinjaman (output yang diamati), dan itu akan membuat model untuk memprediksi siapa yang akan gagal dalam pinjaman.



Manajer produk dapat menggunakan kerangka kerja ini untuk memprediksi hasil bisnis dalam situasi apa pun di mana Anda memiliki data masukan dan keluaran historis:

1 Identifikasi pertanyaan bisnis yang ingin Anda tanyakan.
2 Identifikasi masukan historis.
3 Identifikasi keluaran yang diamati secara historis (mis., Sampel data tentang kapan kondisinya benar dan kapan kondisinya salah).

Misalnya, jika Anda ingin memprediksi siapa yang akan membayar tagihan terlambat, identifikasi input (demografi pelanggan, tagihan) dan output (bayar terlambat atau tidak), dan biarkan machine learning menggunakan data ini untuk membuat model Anda.



Jadi masalah yang perlu diselesaikan oleh perusahaan untuk membuat model prediktif adalah mengidentifikasi sampel data ketika kondisi (churn) benar dan ketika kondisi (churn) salah dan meneruskan data ini ke algoritma prediksi untuk membuat model. Itu adalah cara sederhana untuk melihat hasil prediksi.

Inilah yang sebagai manajer produk dapat menemukan cara untuk memperlakukan data bisnis Anda sebagai aset keuangan. Rumuskan pertanyaan yang memiliki nilai bisnis yang baik, kumpulkan data (historis) yang memiliki sampel hasil / kondisi positif dan negatif di masa lalu dan arahkan algoritme ke data Anda sehingga dapat mempelajari aturan yang kuat yang dapat digunakan untuk memprediksi hasil bisnis di masa depan.

Tabel 1.1 Perbedaan Kecerdasan Buatan dengan Kecerdasan Alami

KECERDASAN BUATAN	KECERDASAN ALAMI
1. Bersifat permanen 2. Lebih mudah diduplikasi dan disebarkan 3. lebih murah 4. Konsisten 5. dapat didokumentasi 6. Lebihcepat 7. dapat mengerjakan pekerjaan lebih baik	1. Cepat mengalami perubahan 2. Proses transfer dari manusia satu kelainnya membutuhkan proses yang lama 3. lebih mahal karena tidak jarang harus mendatangkan orang untuk suatu pekerjaan 4. sering berubah-ubah (sifat manusia) 5. sulit direproduksi 6. Lebih lambat 7. sering kali kurang teliti

1.4 Bidang-bidang Aplikasi AI

Kecerdasan buatan mengambil peran penting dalam penelitian ilmu komputer, manajemen dan wilayah operasional. Kecerdasan adalah istilah yang dikenal sebagai kemahiran tentang subjek dan perintah Anda dan pengetahuan untuk memecahkan masalah yang kompleks. Di tahun-tahun mendatang, mesin kecerdasan buatan akan menggantikannya keterampilan manusia di banyak bidang. AI bekerja pada alat komputasi yang mampu bekerja untuk kecerdasan manusiayang memberi respon, memahami bahasa manusia, mengumpulkan data, dan terhubung dengan objek. Buatan Intelijen adalah istilah yang mirip seperti psikologi tetapi ditambahkan istilah komputasi dari ilmu komputer karena sifatnya penekanan pada persepsi, penalaran dan tindakan. Itu membuat perspektif mesin dan lebih berguna. Teknologi AI menawarkan banyak aplikasi dengan manfaat praktis yang nyata. Kecerdasan Buatan terutama dalam dua istilah Wajah dan Pengenalan Suara, kedua istilah tersebut umumnya digunakan secara bergantian di bidang-bidang seperti robotika. Kecerdasan buatan dapat melakukan tugas-tugas tertentu lebih cepat dan lebih baik daripada manusia. AI melakukan pengujian untuk kembangkan untuk menguji apakah mesin tertentu dapat berpikir atau tidak. Tes ini melibatkan manusia interogator yang menguji dengan manusia dan mesin untuk memeriksa atau menguji siapa manusia dan mana yang mesin.

1.4.1 Bidang Kecerdasan Buatan :

1. Pengenalan ucapan : Kemampuan untuk "memahami" dan menanggapi bahasa alami. Ini mengubah ucapan bahasa ke bentuk tertulis.
2. Sistem pembelajaran dan adaptif : Teknologi untuk menyesuaikan perilaku berdasarkan pengalaman sebelumnya, dan untuk memperbaiki aturan umum tentang dunia berdasarkan pengalaman seperti itu.
3. Pemecahan Masalah : Kapasitas untuk membangun masalah dalam representasi yang sesuai untuk merencanakan solusinya dan untuk temukan kapan informasi baru dibutuhkan dan bagaimana mendapatkannya.
4. Persepsi (visual) : Kapasitas untuk menemukan adegan penginderaan dengan menghubungkannya dengan model internal yang ditampilkan pengakuan menjadi "pengetahuan dunia." Hasil dari analisis ini adalah seperangkat hubungan yang terstruktur antar entitas di tempat kejadian.
5. Modeling: Kemampuan untuk meningkatkan presentasi dan protokol transformasi yang dapat diimplementasikantemukan perilaku dan hubungan di beberapa kumpulan entitas.
6. Robot : Robot adalah istilah yang digunakan untuk menangani objek dengan cara mempersepsikan, mengambil, memindahkan, memodifikasi secara fisik atau memiliki efek membebaskan tenaga kerja dari melakukan pekerjaan berulang tanpa merasa bosan, terganggu, atau kelelahan.
7. Biometrik : Biometrik didasarkan pada pengenalan manusia mungkin persimpangan fisik atau perilaku. Alat biometrik yang digunakan untuk menemukan administrasi dan pengendalian. Ini juga digunakan untuk mencari individu dalam kelompok yang berada di bawah pengawasan. Saat ini digunakan dalam riset pasar.

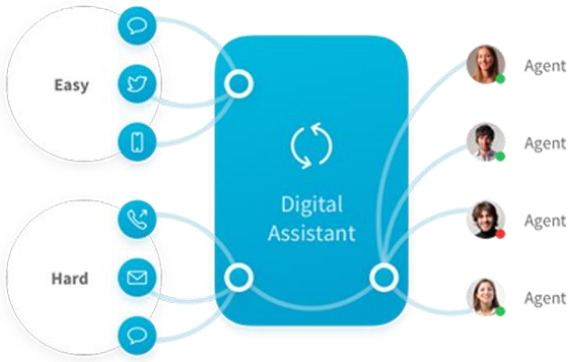
1.4.2 Aplikasi Kecerdasan Buatan

1. Automated customer support

Sekarang ini sulit untuk menemukan toko online yang tidak menawarkan dukungan pelanggan berteknologi tinggi, sebelumnya dilakukan melalui email atau telepon. Saluran dukungan tradisional membebani bisnis dengan sejumlah besar uang dan pemborosan sumber daya manusia dapat diarahkan ke tugas yang lebih cerdas dan kreatif sebaliknya.

Artificial Intelligence memberikan respons untuk pertanyaan sederhana seperti memberikan status pesanan Anda dan menemukan sebuah produk tertentu berdasarkan deskripsi Anda sesuai orang lain.
Pengalaman belanja online telah sangat ditingkatkan oleh Chatbots karena alasan berikut:

a. Mereka menawarkan pemutaran ulang cepat dibandingkan dengan asisten manusia, ini mengurangi waktu.
b. Chatbots juga membantu dalam peluang perdagangan.

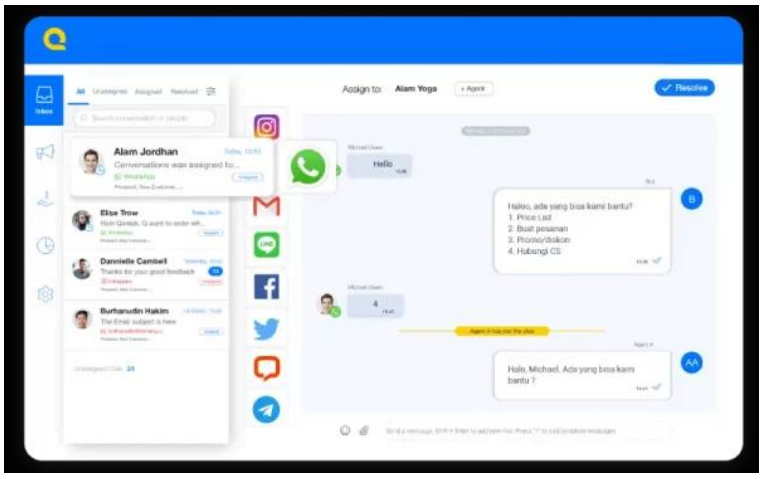


Gambar 1.10 Layanan Pelanggan Otomatis: Dasar-dasar untuk Dukungan yang Kuat

2. Travel and navigation

Dalam industri transportasi, AI menjadi salah satu kunci penting dalam industri ini. Untuk memberikan rute yang akurat untuk bekerja atau membuat pengaturan perjalanan kecerdasan buatan selalu membantu orang dalam kehidupan sehari-hari. Banyak orang mengulas tip perjalanan dan juga memesan perjalanan di perangkat ini, dengan bantuan asisten perjalanan AI. Chatbots juga digunakan untuk industri perjalanan untuk memberikan layanan seperti interaksi dengan pelanggan untuk perjalanan pemberitahuan, detail tentang pemesanan, dan memberikan respon cepat kepada pengguna.

Dalam istilah transportasi, Anda dapat mempertimbangkan pemetaan validasi AI Google Maps, mengidentifikasi informasi jalan dan menggunakannya untuk algoritme untuk mengidentifikasi rute tercepat untuk sepeda, mobil, dan juga berjalan kaki. Sekarang hari Al ada di mana-mana bisa berupa bidang bisnis, pendidikan, teknis atau nonteknis.



Gambar 1.11 Pelayanan otomatis dengan chatbot di seluruh channel bisnis

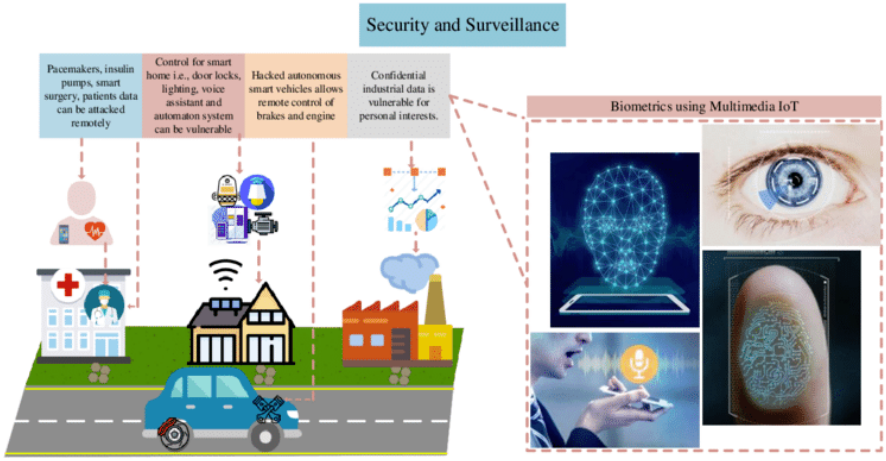
3. Smart home devices

Peningkatan penggunaan AI telah memperkuat istilah "rumah pintar". Perangkat rumah pintar yang Anda gunakan AI teknologi untuk mengadopsi perilaku Anda sehingga mereka dapat mengubah pengaturan mereka dengan cepat untuk menikmati pengalaman semulus bisa jadi. Asisten suara pintar sedang mengerjakan perangkat pintar ini. Butuh beberapa saat untuk melihat rumah berbasis teknologi AI yang terdefinisi dengan baik. AI mampu merespon pilihan kita dalam kehidupan nyata, dibutuhkan langkah ke depan membawa teknologi ini lebih dekat ke dunia nyata. Ada juga teknologi lampu pintar yang bisa mengubah daya dan warna lampu berdasarkan waktu.

Gambar 1.12 Smart Home System

4. Security and surveillance

Saat berdebat tentang aspek etika dalam menerapkan sistem pengawasan yang luas, maka tidak dapat menghindari fakta itu sedang diterapkan. Mungkin tidak semua orang dapat secara konsisten memantau semua saluran dengan data datang dari sejumlah besar kamera pada saat yang sama, tetapi teknologi AI memungkinkannya. Teknologi AI yang sebagian besar bekerja pada pengenalan suara dan teknologi berbasis pengenalan wajah semakin meningkat mempersonalisasi pengalaman. Teknologi pengolahan citra menerapkan ilmu data dengan meningkatkan kecerdasan buatan.



Gambar 1.13 Teknologi Pemananan dan Pengawasan

5. Artificial Intelligence in Healthcare

Pusat perawatan kesehatan menggunakan teknologi pembelajaran mesin untuk membuat pemulihan penyakit lebih baik dan lebih cepat daripada manusia. AI adalah studi yang dibuat untuk meniru kecerdasan manusia menjadi TIK yang dapat membantu dokter dan pasien dengan cara berikut:

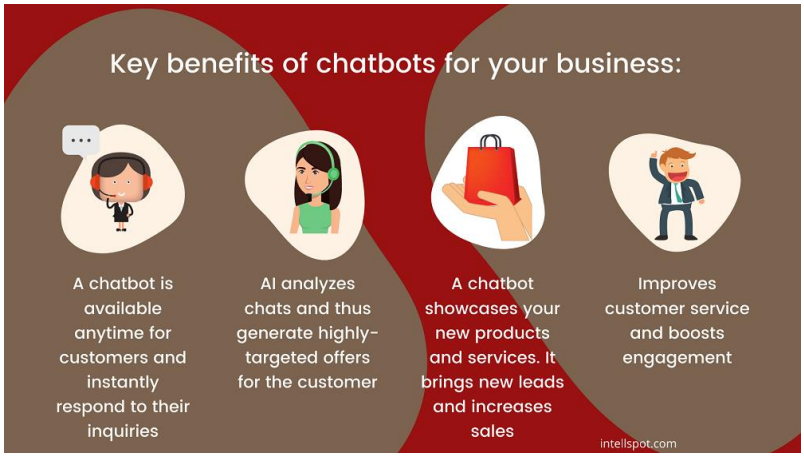
Dengan menyediakan laboratorium untuk berbagai eksperimen, representasi dan penganalisaan informasi medis.Dengan datang dengan alat baru untuk memberikan dukungan untuk pengambilan keputusan dan penelitian Dengan menambahkan aktivitas dalam ilmu kedokteran, perangkat lunak dan kognitif Dalam perawatan kesehatan, kecerdasan buatan telah bekerja sebagai pengubah permainan, itu mengembangkan setiap industri secara efektif. Mungkin catatan pribadi pasien yang aman dari penjahat dunia maya untuk bekerja sebagai bantuan dalam operasi - AI dikenali di mana-mana.AI memfasilitasi

para dokter untuk mengurangi jadwal mereka, membebaskan waktu dan biaya dengan merampingkan proses dan membuka jalan baru bagi industri.

Gambar 1.14 Teknologi Peralatan Kedokteran

6. Artificial Intelligence in business

Teknologi robotik digunakan untuk melakukan tugas-tugas yang sangat kritis yang biasanya diselesaikan oleh manusia. Platform manajemen hubungan pelanggan untuk mengungkap informasi tentang cara melayani pelanggan dengan lebih baik. Chatbots telah terlibat dalam situs web dan perusahaan elektronik untuk memberikan layanan yang cepat dan lancar kepada pelanggan.Koordinasi industri keuangan dan teknologi AI sangat cocok. Sektor keuangan semakin bekerja pada pelaporan real-time yang sebenarnya dan memproses sejumlah besar data penting untuk membuat hasil yang penting.Di semua bagian ini di mana kecerdasan buatan memungkinkan sistem menjadi unggul. AI memberikan data yang akurat dan efisien, chatbots, komputerisasi, dll adalah bagian dari proses ini.



Gambar 1.15 Manfaat utama chatbot untuk bisnis

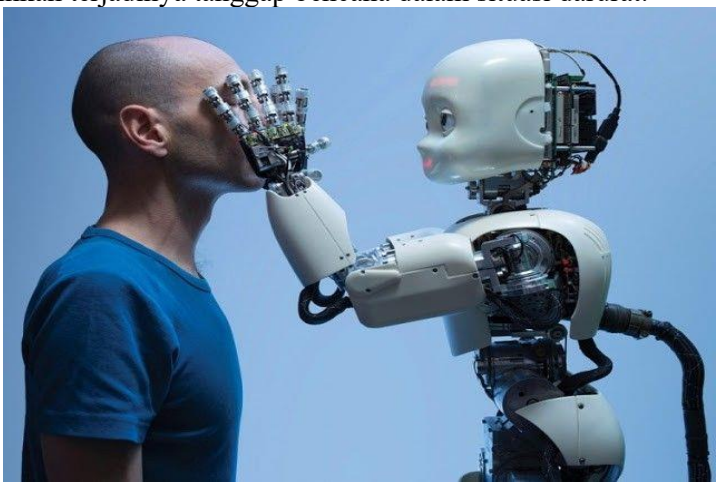
7. AI in education

Dalam penilaian otomatis, memberi tutor lebih banyak waktu, itu bermanfaat untuk menghitung siswa, menyesuaikan dengan keinginan mereka dan juga mendukung mereka untuk bekerja sesuai bidang perhatian mereka.

Gambar 1.16 Penggunaan AI dan VR di Sektor Pendidikan

8. AI for robotics

Robotika terutama dapat digunakan untuk merawat orang yang berusia di atas dan juga memungkinkan kemandirian yang panjang. Bahkan akan menurunkan angka kecelakaan lalu lintas dan kematian, juga memungkinkan terjadinya tanggap bencana dalam situasi darurat.



Gambar 1.17 Manusia Robot

Soal :

1. Carilah contoh Aplikasi yang masuk dalam bidang kecerdasan buatan.
2. Dari soal nomer 1, sebutkan apa nama aplikasi tersebut ?
3. Dari soal nomer 1, Jelaskan cara kerja dari aplikaisi tersebut

BAB 2
Masalah Dan Metode Pemecahan Masalah

2.1 Representasi Masalah

Representasi state-space dari suatu masalah menggunakan dua jenis entitas: state, yang merupakan struktur data yang memberikan "snapshots" dari kondisi masalah pada setiap tahap penyelesaiannya, dan operator, yang berarti untuk mengubah masalah dari satu state. ke yang lain.
Contoh sederhana dari representasi state-space adalah teka-teki sederhana yang terkenal yang disebut 8-puzzle. Teka-teki 8 adalah baki persegi yang berisi 8 ubin persegi dengan ukuran yang sama, bernomor 1 sampai 8. Ruang untuk ubin ke-9 kosong (lihat Gambar 2.1).

2	1	6
4		8
7	5	3

Gambar 2.1: Sebuah 8-teka-teki.

Sebuah ubin dapat dipindahkan dengan menggesernya secara vertikal atau horizontal ke dalam kotak kosong. Masalahnya adalah untuk mengubah satu konfigurasi ubin, katakanlah dari Gambar 2.1, menjadi konfigurasi ubin lain yang diberikan, katakanlah dari Gambar 2.2.

1	2	3
8		4
7	6	5

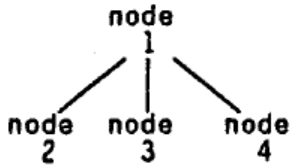
Gambar 2.2: Konfigurasi solusi dari 8-puzzle.

Status adalah konfigurasi ubin tertentu; setiap state dapat diwakili oleh matriks 3 x 3, mirip dengan Gambar 2.1 dan 2.2. Operator, sesuai dengan kemungkinan gerakan, dapat didefinisikan dengan operator terpisah untuk masing-masing ubin 1 hingga 8. Namun, definisi yang lebih ringkas dimungkinkan dengan melihat bujur sangkar kosong sebagai objek yang akan dipindahkan dan menyatakan operator dalam hal pergerakan bujur sangkar ini. Dalam formulasi ini, hanya empat operator yang digunakan:

- “Up” pindahkan yang kosong ke atas satu kotak
- “Down” pindahkan yang kosong ke bawah satu kotak
- “Left” pindahkan yang kosong ke kiri satu kotak
- “right” pindahkan yang kosong ke kanan satu kotak

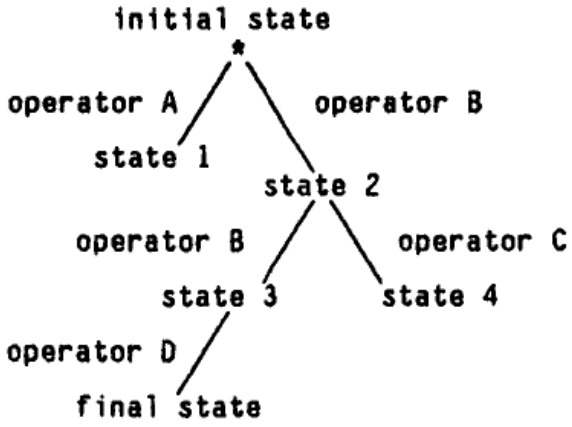
Operator mungkin tidak dapat diterapkan di bagian tertentu, seperti saat akan memindahkan blanko di luar baki ubin. Himpunan semua keadaan yang dapat dicapai dari suatu masalah sering disebut ruang keadaannya. Teka-teki 8-, misalnya, memiliki ruang keadaan berukuran 9, karena ada 9 konfigurasi ubin tetapi hanya setengah dari jumlah ini yang dapat dicapai dari konfigurasi awal yang diberikan. Ini hanya terjadi pada 181.440 kemungkinan keadaan. Empat operator yang didefinisikan untuk 8-puzzle membentuk satu set fungsi parsial pada ruang keadaan: Setiap operator, jika berlaku untuk keadaan tertentu sama sekali, mengembalikan tepat satu keadaan baru sebagai hasilnya. Dalam masalah yang lebih kompleks, bagaimanapun, operator sering mengandung variabel. Jika, untuk keadaan dan operator tertentu, variabel dapat diinstansiasi dalam lebih dari satu cara, maka setiap instantiasi menghasilkan satu keadaan baru, dan operator masalah, jika dianggap sebagai fungsi pendefinisi, lebih tepat disebut operator.
Spesifikasi lengkap dari masalah state-space memiliki tiga komponen. Salah satunya adalah himpunan 0 operator atau operator chemata. Selain itu, seseorang harus mendefinisikan himpunan S dari satu atau

lebih keadaan awal dan menemukan predikat yang mendefinisikan himpunan G keadaan tujuan. Masalah state-space adalah triple (S, 0, G). Solusi untuk masalah ini adalah urutan aplikasi operator yang terbatas yang mengubah keadaan awal menjadi keadaan tujuan.
Ruang keadaan dapat diperlakukan sebagai graf berarah yang simpulnya adalah keadaan dan busurnya adalah operator yang mengubah satu keadaan ke keadaan lain. Sebagai contoh, jika keadaan 1 adalah keadaan dimana salah satu dari tiga operator dapat diterapkan, mengubahnya menjadi keadaan 2, 3, atau 4, maka grafik yang sesuai akan seperti Pada Gambar 2.3. Node 2, 3, dan 4 disebut penerus dari node 1.



Gambar 2.3: Busur yang diarahkan

Dalam notasi graf, solusi untuk masalah state-space adalah jalur dari node awal ke node tujuan. Pada Gambar 2.4, satu solusi akan menjadi aplikasi operator B dua kali, diikuti oleh operator D, untuk mencapai simpul tujuan yang ditunjukkan atau keadaan akhir. Mungkin ada keadaan akhir lainnya dan beberapa cara untuk mencapai keadaan akhir tertentu.

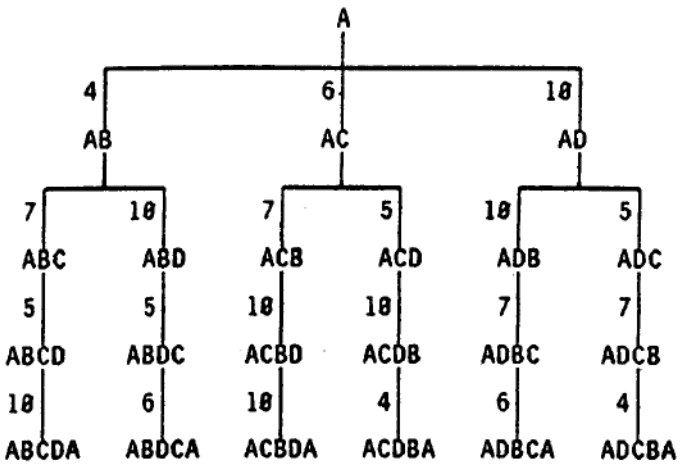


Gambar 2.4: Grafik state-space

Variasi umum pada masalah state-space membutuhkan pencarian bukan sembarang jalur tetapi salah satu biaya minimum antara node awal dan node tujuan. Dalam hal ini, setiap busur grafik diberi label dengan biayanya. Contohnya adalah masalah travelling-salesman: Diberikan sejumlah kota yang akan dikunjungi dan jarak tempuh antara setiap pasangan kota, carilah jarak tempuh perjalanan minimum yang dimulai dan diakhiri di kota A yang mengunjungi masing-masing kota lainnya tepat satu kali. Contoh grafik jarak tempuh dan grafik ruang-negara bagian yang sesuai ditunjukkan pada Gambar 2.5. Karena jalur yang berbeda ke kota yang sama mewakili solusi parsial yang berbeda, setiap negara bagian diidentifikasi tidak hanya sebagai nama kota tetapi sebagai daftar kota yang dikunjungi sejauh ini.

	A	B	C	D
A	–	4	6	10
B		–	7	10
C			–	5
D				–

Bagan jarak tempuh



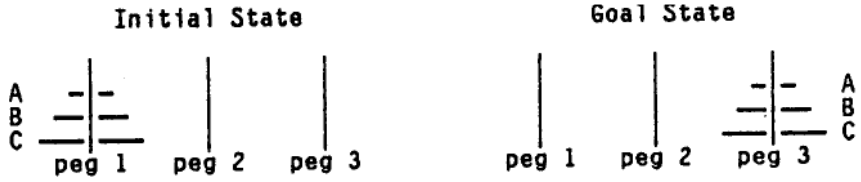
State-space graph

Gambar 2.5: sebuah traveling-salesman problem.

Solusi yang diinginkan adalah ABDCA, atau pembalikannya, dengan total jarak tempuh 25. (Dua tingkat terbawah dari grafik dapat dihilangkan, karena jarak tempuh setiap tur dari n kota ditentukan oleh n-1 kota pertama yang dipilih untuk menjadi dikunjungi.)
Karena graf state-space biasanya terlalu besar untuk direpresentasikan secara eksplisit, masalah pencarian solusi menjadi salah satu pembangkitan graf yang cukup untuk memuat jalur solusi yang diinginkan.

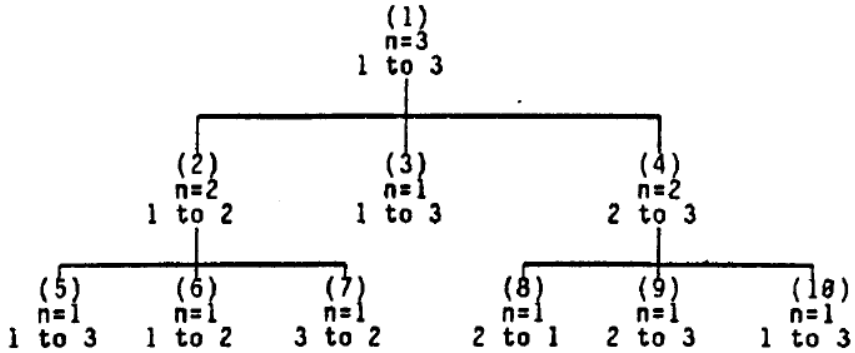
2.2. Representasi pengurangan masalah

Sering dibedakan dari representasi state-space masalah adalah teknik yang disebut representasi pengurangan masalah. Dalam pendekatan pengurangan masalah, struktur data utama adalah deskripsi masalah atau tujuan. Deskripsi masalah awal diberikan; itu diselesaikan dengan urutan transformasi yang pada akhirnya mengubahnya menjadi satu set sub masalah yang solusinya segera. Transformasi yang diizinkan didefinisikan sebagai operator. Seorang operator dapat mengubah satu masalah menjadi beberapa sub masalah; untuk memecahkan yang pertama, semua sub masalah harus diselesaikan. Selain itu, beberapa operator yang berbeda mungkin dapat diterapkan pada satu masalah, atau operator yang sama dapat diterapkan dalam beberapa cara yang berbeda. Dalam hal ini cukup untuk memecahkan sub masalah yang dihasilkan oleh salah satu aplikasi operator. Masalah yang penyelesaiannya segera disebut masalah primitif. Dengan demikian, representasi masalah menggunakan reduksi masalah didefinisikan oleh rangkap tiga yang terdiri dari:
(a) deskripsi masalah awal,
(b) seperangkat operator untuk mengubah masalah menjadi sub masalah, dan
(c) satu set deskripsi masalah primitif.
Penalaran berjalan mundur dari tujuan awal
Contoh :
Contoh yang cocok untuk representasi pengurangan masalah adalah teka-teki Menara Hanoi yang terkenal. Dalam satu versi umum ada tiga disk, A, B, dan C, dengan ukuran bertingkat. Ada juga tiga pasak, 1, 2, dan 3. Awalnya disk ditumpuk di pasak 1 , dengan A, yang terkecil, di atas dan C, yang terbesar, di bawah. Masalahnya adalah mentransfer tumpukan ke pasak 3, seperti pada Gambar 2.6, mengingat (a) hanya satu disk yang dapat dipindahkan pada satu waktu, dan (b) tidak ada disk yang dapat ditempatkan di atas disk yang lebih kecil.



Gambar 2.6: Teka-teki Menara Hanoi.

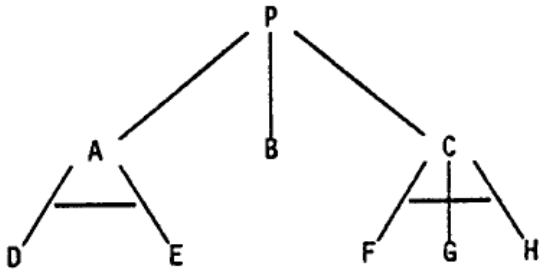
Hanya satu operator yang perlu digunakan dalam solusi: Mengingat pasak i, j, dan k yang berbeda, masalah memindahkan tumpukan ukuran n > 1 dari pasak i ke pasak k dapat diganti dengan tiga masalah:
(a) memindahkan setumpuk ukuran n-1 dari i ke J,
(b) memindahkan setumpuk ukuran 1 dari i ke k, dan
(c) memindahkan setumpuk ukuran n-1 dari j ke k.
Satu-satunya masalah primitif adalah memindahkan satu disk dari satu pasak ke pasak lain, asalkan tidak ada disk yang lebih kecil di pasak penerima. Jika ada disk yang lebih kecil, masalah ini tidak akan terpecahkan (mengingat definisi dari satu-satunya operator yang tersedia).
Setiap deskripsi masalah sekarang dapat diberikan dengan menentukan ukuran n tumpukan yang akan dipindahkan, jumlah pasak pengirim, dan jumlah pasak penerima. Masalah asli, memindahkan tumpukan tiga disk dari pasak 1 ke pasak 3, kemudian akan direpresentasikan sebagai (n = 3, 1 ke 3), dan transformasi masalah asli ke masalah primitif dapat diwakili oleh pohon:



Gambar 2.7: Solusi teka-teki Menara Hanoi.

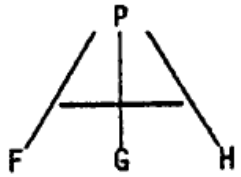
Ada dua kemungkinan urutan operator yang mengubah masalah asli menjadi masalah primitif: Terapkan operator ke node 1, lalu node 2, dan kemudian node 4; atau terapkan operator ke simpul 1, lalu simpul 4, dan kemudian simpul 2. Karena simpul 3 adalah masalah primitif, maka tidak perlu diperhatikan lebih lanjut. Node 2 merepresentasikan subproblem pemindahan dua disk teratas pada pasak 1 ke pasak 2. Submasalah ini diselesaikan dengan memperluasnya ke masalah primitif pada node (5), (6), dan (7) yang diselesaikan dengan memindahkan disk terkecil ke pasak 3, memindahkan disk tengah ke pasak 2, dan akhirnya meletakkan disk kecil kembali di atas disk tengah.
Urutan operator yang akan diterapkan harus dibedakan dari urutan tindakan yang akan dilakukan untuk mencapai tujuan. Dalam contoh Menara Hanoi, tindakannya adalah gerakan disk yang sebenarnya. Urutan ini diberikan oleh simpul terminal pohon, dibaca dari kiri ke kanan. Apakah dianggap penting atau tidak untuk merakit urutan tindakan seperti itu tergantung pada domain masalah tertentu.
Dalam contoh di atas, sebuah pohon digunakan untuk menampilkan solusi pengurangan masalah pada teka-teki Menara Hanoi. Notasi pohon harus digeneralisasikan jika ingin mewakili berbagai macam situasi yang mungkin terjadi dalam pengurangan masalah. Notasi umum untuk pengurangan masalah ini disebut grafik ANDIOR. Menurut satu formulasi umum (Nilsson, 1971), grafik AND/OR dibangun menurut aturan berikut:

1. Setiap node mewakili baik satu masalah atau satu set masalah yang harus dipecahkan. Grafik berisi simpul awal yang sesuai dengan masalah asli.
2. 2. Sebuah node yang mewakili masalah primitif, disebut terminal node, tidak memiliki keturunan.
3. 3. Untuk setiap kemungkinan penerapan operator pada masalah P, mentransformasikannya menjadi himpunan submasalah, terdapat busur berarah dari P ke simpul yang mewakili himpunan submasalah yang dihasilkan. Sebagai contoh, Gambar 3 mengilustrasikan pengurangan P menjadi tiga himpunan submasalah yang berbeda: A, B, dan C. Karena P dapat diselesaikan jika salah satu himpunan A, B, atau C dapat diselesaikan, A, B, dan C adalah disebut OR node.



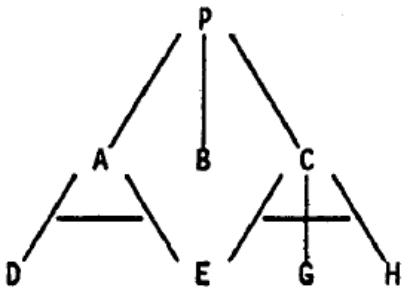
Gambar 2.8: Pohon AND/OR

4. Gambar 2.8 mengilustrasikan lebih lanjut komposisi himpunan A, B, dan C: A = {D, E}, B terdiri dari satu soal (tidak disebutkan namanya), dan C = {F, G, H>. Secara umum, untuk setiap node yang mewakili satu set dua atau lebih submasalah, ada busur yang diarahkan dari node untuk himpunan ke node individu untuk setiap submasalah. Karena satu set subproblem hanya dapat diselesaikan jika semua anggotanya dapat diselesaikan, node subproblem disebut AND node. Untuk membedakannya dari node OR, busur yang mengarah ke penerus AND node dari induk yang sama dihubungkan dengan garis horizontal.
5. Penyederhanaan graf yang dihasilkan oleh aturan 3 dan 4 dapat dibuat dalam kasus khusus di mana hanya satu penerapan operator yang mungkin untuk masalah P dan di mana operator ini menghasilkan himpunan lebih dari satu submasalah. Seperti yang diilustrasikan Gambar 2.9, simpul OR perantara yang mewakili himpunan submasalah kemudian dapat dihilangkan:



Gambar 2.9: Pohon AND/OR tree dengan operator di problem P.

Contoh lain dari konstruksi ini diberikan pada Gambar 2.7
Pada gambar di atas, setiap node mewakili masalah yang berbeda atau serangkaian masalah. Karena setiap simpul kecuali simpul awal hanya memiliki satu induk, grafiknya sebenarnya adalah pohon ANDIOR. Sebagai variasi pada Gambar 2.8, asumsikan bahwa masalah A dapat direduksi menjadi D dan E; dan masalah C, ke E, G, dan H. Kemudian E dapat diwakili oleh dua node yang berbeda, atau oleh satu node seperti yang ditunjukkan pada Gambar 2.10. Pilihan membuat perbedaan dalam algoritma pencarian yang akan dibahas nanti dalam bab ini. Misalnya, jika simpul E pada gilirannya dapat direduksi menjadi C, representasi grafik umum hanya menambahkan busur berarah lain ke Gambar 2.10, tetapi pohon yang sesuai menjadi tak terbatas.



Gambar 2.10: Grafik AND/OR.

Konstruksi yang dibahas sejauh ini menyangkut grafik yang menggambarkan seluruh ruang pencarian masalah. Untuk menemukan solusi untuk masalah awal, kita hanya perlu membangun grafik yang cukup untuk menunjukkan bahwa node awal dapat diselesaikan. Subgraf semacam itu disebut graf solusi atau, dalam kasus pohon AND/OR yang lebih terbatas, pohon solusi. Aturan berikut berlaku:
Sebuah node dapat dipecahkan jika:

a. itu adalah simpul terminal (masalah primitif);
b. itu adalah simpul nonterminal yang penerusnya adalah simpul AND yang semuanya dapat dipecahkan; atau
c. itu adalah simpul nonterminal yang penerusnya adalah simpul OR dan setidaknya salah satunya dapat dipecahkan.

Demikian pula, sebuah simpul tidak dapat dipecahkan jika:

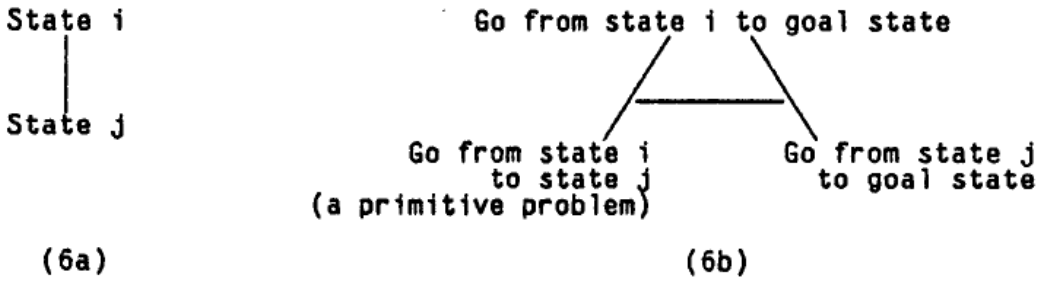
a. itu adalah simpul nonterminal yang tidak memiliki penerus (nonprimitif
b. masalah yang tidak dapat diterapkan oleh operator);
c. itu adalah simpul nonterminal yang penerusnya adalah simpul AND dan setidaknya salah satunya tidak dapat dipecahkan; atau
d. itu adalah simpul nonterminal yang penerusnya adalah simpul OR dan semuanya tidak dapat dipecahkan.

Metode pencarian grafik AND/OR untuk solusi tersebut dibahas dalam Artikel CS dan C4.

Hubungan antara Pengurangan Masalah dan Representasi Ruang Representations

Beberapa hubungan umum yang menarik dapat ditemukan antara pengurangan masalah dan representasi ruang-negara. Pertama, meskipun satu representasi sering tampak lebih alami untuk masalah yang diberikan, seringkali definisi masalah dapat disusun kembali sehingga menggunakan bentuk lain. Misalnya, teka-teki Menara Hanoi juga dapat diselesaikan dengan pencarian ruang-negara menggunakan operator yang menggerakkan satu disk dan yang mewakili semua langkah legal dalam konfigurasi tertentu. Dibandingkan dengan representasi pengurangan masalah, yang sebenarnya memberikan algoritma untuk memecahkan teka-teki, representasi state-space akan menjadi representasi yang buruk karena menyisakan ruang untuk mencari jalur panjang yang tidak perlu. Kedua, adalah mungkin untuk menerjemahkan secara mekanis antara representasi state-space dan representasi pengurangan masalah tanpa perubahan mendasar dalam cara melihat masalah. Cara membuat terjemahan semacam itu dapat memberikan wawasan yang bermanfaat ke dalam banyak program pencarian di mana konsep ruang keadaan dan representasi pengurangan masalah tampaknya bercampur. Beberapa skema terjemahan dijelaskan di bawah ini. (Beberapa pembaca mungkin ingin melewatkan materi berikut pada bacaan pertama.)

Ruang negara untuk pengurangan masalah. Dua pendekatan menyarankan diri mereka sendiri untuk menerjemahkan representasi state-space ke representasi pengurangan masalah. Dalam satu, grafik state-space dipahami sebagai grafik AND/OR yang hanya berisi node OR. Setiap status dari versi state-space sesuai dengan masalah untuk berpindah dari state tersebut ke state tujuan; dan keadaan tujuan dari ruang keadaan menjadi masalah primitif untuk berpindah dari keadaan tujuan itu ke dirinya sendiri. Dengan kata lain, struktur data yang mewakili keadaan hanya ditafsirkan ulang sebagai representasi deskripsi masalah, di mana masalah terdiri dari informasi keadaan bersama dengan tujuan implisit. Sebagai alternatif, ada sedikit variasi dari pendekatan pertama yang memerlukan pendefinisian ulang operator representasi state-space. Setiap operator tersebut, mengambil keadaan i ke keadaan j, menjadi operator yang berlaku untuk masalah mendapatkan dari keadaan i ke keadaan tujuan. Efeknya adalah mereduksi masalah menjadi sepasang submasalah: (a) berpindah dari keadaan i ke keadaan j (masalah primitif), dan (b) beralih dari keadaan j ke keadaan tujuan. Gambar 2.11 mengilustrasikan korespondensi ini.



Gambar 2.11: (6a) Bagian dari pohon state-space; (6b) yang sesuai bagian dari pohon AND/OR (pengurangan masalah).

Reduksi masalah ke ruang keadaan. Terjemahan dari representasi pengurangan masalah ke representasi ruang keadaan sedikit lebih kompleks, dengan asumsi bahwa operator pengurangan masalah sebenarnya menghasilkan node AND. Masalah awal representasi pengurangan masalah dapat dipahami memiliki dua komponen: (a) deskripsi tujuan yang ingin dicapai, seperti yang dibahas di awal artikel ini, dan (b) deskripsi keadaan awal dunia. Komponen-komponen ini akan dilambangkan masing-masing c__ dan sO. Beberapa contohnya adalah
<_0 = teorema yang akan dibuktikan, dan sO = aksioma untuk membuktikannya;
gO = konfigurasi objek yang ingin dicapai, dan sO = konfigurasi yang ada.
Setiap state S dari representasi state-space yang sesuai adalah pasangan yang terdiri dari setumpuk tujuan (gi, ..., gO) yang akan dicapai dan state saat ini di dunia. Jadi, state awal SO dari representasi state-space adalah SO = ((gO), sO). Keadaan akhir adalah keadaan di mana tumpukan tujuan yang ingin dicapai telah dikosongkan. Untuk setiap operator pengurangan masalah, memetakan masalah atau tujuan g ke himpunan subtujuan {gm, ..., gn), representasi ruang keadaan memiliki operator yang sesuai memetakan keadaan Sl, di mana S1 = ((gi, .. ., gO), s), ke status S2 di mana {gm, ..., gn} telah ditambahkan ke atas tumpukan gawang (dalam urutan pelaksanaannya, jika relevan), dan keadaan dunia tidak berubah; yaitu, S2 ((gm, ..., gn, gi, ..., gO), s).
Representasi state-space juga membutuhkan operator jenis kedua, yang dapat diterapkan setiap kali tujuan di atas tumpukan mewakili masalah primitif. Fungsinya adalah untuk menghapus masalah primitif itu dari tumpukan dan, pada saat yang sama, mengubah status s untuk mencerminkan solusinya. Dalam teka-teki Menara Hanoi, misalnya, keadaan baru akan mencerminkan posisi yang berubah dari satu cakram. Dalam masalah pembuktian teorema, keadaan baru akan berbeda dari yang lama dengan penambahan satu formula ke yang telah diberikan sebagai aksioma atau ditetapkan dari penyelesaian submasalah sebelumnya. Representasi jenis ini digunakan secara eksplisit dalam program STRIPS Fikes dan Nilsson, yang dijelaskan dalam Pasal D5.

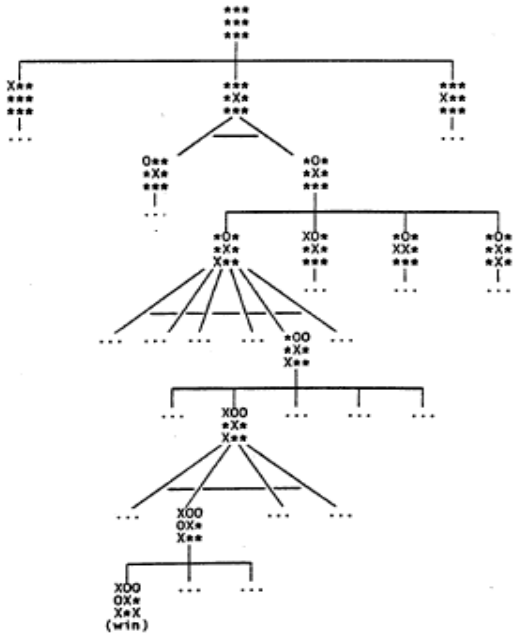
Game Trees
Sebagian besar permainan yang dimainkan oleh program komputer, termasuk catur, catur, go, dan tic-tactoe, memiliki beberapa kesamaan fitur dasar. Ada dua pemain yang bergantian dalam melakukan gerakan. Pada setiap belokan, aturan menentukan gerakan apa yang legal dan efek yang akan ditimbulkan oleh setiap gerakan yang mungkin; tidak ada unsur kebetulan. Berbeda dengan permainan kartu di mana tangan pemain disembunyikan, setiap pemain memiliki informasi lengkap tentang posisi lawannya, termasuk pilihan yang terbuka untuknya dan gerakan yang telah dibuatnya. Permainan dimulai dari keadaan tertentu, seringkali merupakan konfigurasi pria di papan. Itu berakhir dengan kemenangan untuk satu pemain dan kekalahan untuk yang lain, atau mungkin seri.
Pohon permainan yang lengkap adalah representasi dari semua kemungkinan permainan dari permainan tersebut. Node akar adalah keadaan awal, di mana giliran pemain pertama yang bergerak. Penerusnya adalah negara bagian yang bisa dia capai dalam satu gerakan; penerusnya adalah status yang dihasilkan dari kemungkinan balasan pemain lain; dan seterusnya. Status terminal adalah yang mewakili menang, kalah, atau seri. Setiap jalur dari simpul akar ke simpul terminal memberikan permainan lengkap yang berbeda dari permainan.
Perbedaan penting antara pohon permainan dan pohon ruang keadaan adalah bahwa pohon permainan mewakili gerakan dua pemain yang berlawanan, misalnya A dan B. Namun, pohon DAN/ATAU cukup

untuk mencerminkan oposisi ini. Pohon permainan biasanya digambar untuk mewakili hanya satu sudut pandang pemain. Dalam pohon permainan yang diambil dari sudut pandang A, kemungkinan pergerakan A dari posisi tertentu diwakili oleh node OR karena mereka adalah alternatif di bawah kendalinya sendiri. Gerakan yang mungkin dilakukan B sebagai balasannya adalah node AND, karena mereka mewakili kumpulan gerakan yang harus dapat ditanggapi oleh A. Karena pemain bergiliran, OR node dan AND node muncul pada level alternatif dari pohon. Dalam bahasa grafik AND/OR, pohon menampilkan ruang pencarian untuk masalah yang menunjukkan bahwa A bisa menang. Sebuah node yang mewakili kemenangan untuk A sesuai dengan masalah primitif; simpul yang mewakili kemenangan untuk B atau seri, untuk masalah yang belum terpecahkan. Berbeda dengan terminologi grafik AND/OR biasa, kedua jenis node ini akan disebut node terminal.

Sebagai contoh, Gambar 1 menunjukkan sebagian dari pohon permainan ban untuk tic-tac-toe. Pemainnya adalah X dan 0, X memiliki langkah pertama, dan pohon diambil dari sudut pandang X. Posisi dianggap identik jika satu dapat diperoleh dari yang lain dengan rotasi atau refleksi dari grid. Pohon juga dapat digambarkan dari sudut pandang O, meskipun X memiliki langkah pertama. Dalam hal ini, simpul AND akan menjadi simpul OR, dan sebaliknya, dan label "menang" dan "kalah" akan dibalik. Formulasi alternatif pohon permainan, tidak secara eksplisit membedakan antara AND dan OR node, diberikan dalam Artikel Csa, Minimax.

Metode pencarian pohon permainan untuk strategi kemenangan dibahas di Bagian C5. Seperti halnya pencarian di domain lain, sumber kesulitan dalam permainan yang menantang adalah ruang pencarian yang sangat besar. Sebuah pohon permainan lengkap untuk catur, misalnya, yang lebih sulit daripada tic-tac-toe tetapi jauh lebih sederhana daripada catur atau go, telah diperkirakan memiliki sekitar 1040 node nonterminal. Jika seseorang berasumsi bahwa simpul-simpul ini dapat dihasilkan dengan kecepatan 3 miliar per detik, generasi seluruh pohon masih akan membutuhkan sekitar 1021 abad!

Game Trees

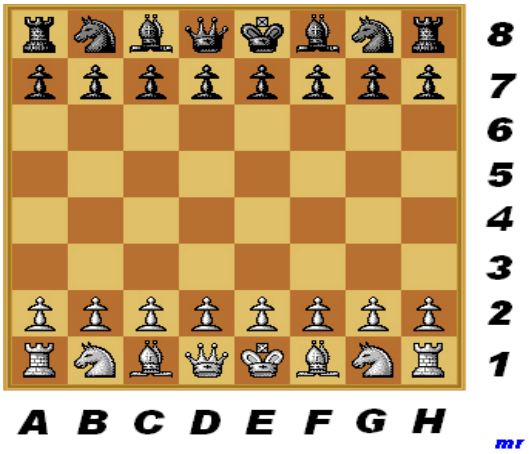


Gambar 2.12. Pohon permainan untuk Tic-tac-toe

2.3 Mendefinisikan Masalah Sebagai Suatu Ruang Keadaan

Misalkan permasalahan yang dihadapi adalah permainan catur, maka harus ditentukan :

1. **posisi awal** pada papan catur posisi awal setiap permainan catur selalu sama, yaitu semua bidak diletakkan di atas papan catur dalam 2 sisi, yaitu kubu putih dan kubu hitam.



Gambar 2.13 Posisi awal dari permainan Catur

2. Aturan – aturan untuk melakukan gerakan

Aturan – aturan ini sangat berguna untuk menentukan gerakan suatu bidak, yaitu melangkah dari satu keadaan ke keadaan lain. Misalkan untuk mempermudah menunjukkan posisi bidak, setiap kotak ditunjukkan dalam huruf (a,b,c,d,e,f,g,h) pada arah horisontal dan angka (1,2,3,4,5,6,7,8) pada arah vertikal. Suatu aturan untuk menggerakkan bidak dari posisi (e,2) ke (e,4) dapat ditunjukkan dengan aturan : if bidak putih pada kotak(e,2), and kotak(e,3) kosong, and kotak(e,4) kosong then gerakkan bidak dari (e,2) ke (e,4).

3. T**ujuan (goal)**

Tujuan yang ingin dicapai adalah posisi pada papan catur yang menunjukkan kemenangan seseorang terhadap lawannya. Kemenangan ini ditandai dengan posisi raja yang sudah tidak dapat bergerak lagi.

Gambar 2.14 Posisi Akhir dari permainan Catur

Mendeskripsikan masalah dengan baik harus :

1. Mendefinisikan suatu ruang keadaan (state space)
2. Menetapkan satu atau lebih keadaan awal (initial state)
3. Menetapkan satu atau lebih tujuan (goal state)
4. Menetapkan kumpulan aturan

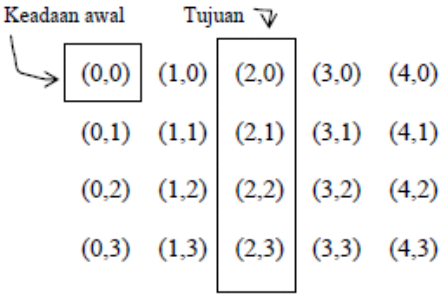
Contoh Permasalahan :

Ada 2 ember masing-masing berkapasitas 4 galon (ember A) dan 3 galon (ember B). Ada pompa air yg akan digunakan untuk mengisi air pada ember tersebut. Bagaimana dapat mengisi tepat 2 galon air ke dalam ember berkapasitas 4 galon?

Penyelesaian :

1. **Identifikasi ruang keadaan (state space)**

Permasalahan ini dapat digambarkan sebagai himpunan pasangan bilangan bulat :
x = jumlah air yg diisikan ke ember 4 galon (ember A)
y = jumlah air yg diisikan ke ember 3 galon (ember B)
Ruang keadaan = (x,y) sedemikian hingga x $\in$ {0,1,2,3,4} dan y $\in${0,1,2,3}
2. **Keadaan awal & tujuan**
Keadaan awal : kedua ember kosong = (0,0)
Tujuan : ember 4 galon berisi 2 galon air = (2,n) dengan sembarang n
3. **Keadaan ember**
Keadaan ember bisa digambarkan sebagai berikut



Gambar 2.15 Posisi Keadaan awal dan tujuan

4. Aturan-aturan Diasumsikan kita dapat mengisi ember air itu dari pompa air, membuang air dari ember ke luar, menuangkan air dari ember yang satu ke ember yang lain.
Kita buat beberapa aturan-aturan yang dapat digambarkan sebagai berikut :

Tabel 2.1 Tabel Aturan-aturan yang dipakai

Aturan ke	Jika	Maka
1	(x,y) x < 4	(4,y) Isi Ember A
2	(x,y) y < 3	(x,3) Isi Ember B
3	(x,y) x > 0	(x-d, y) Tuan sebagian air keluar dari ember A
4	(x,y) y > 0	(x, y-d) Tuan sebagian air keluar dari ember B
5	(x,y) x > 0	(0,y) Kosongkan ember A dengan membuang airnya
6	(x,y) y > 0	(x,0) Kosongkan ember B dengan membuang airnya
7	(x,y) x + y ≥ 4 dan y > 0	(4,y – (4-x)) Tuan air dari ember B ke ember A sampai ember A penuh
8	(x,y) x + y ≥ 3 dan x > 0	(x- (3-y), 3) Tuan air dari ember A ke ember B sampai ember B penuh
9	(x,y) x + y ≤ 4 dan y > 0	(x+y,0) Tuan seluruh air dari ember B ke ember A
10	(x,y) x + y ≤ 3 dan x > 0	(0, x+y) Tuan seluruh air dari ember A ke ember B
11	(0,2)	(2,0) Tuan 2 galon air dari ember B ke ember A

Solusi yang ditemukan :
Solusi 1

Isi Ember A	Isi Ember B	Aturan yang dipakai
0	0	1
4	0	8
1	3	6
1	0	10
0	1	1
4	1	8
2	3	Solusi

Solusi 2

Isi Ember A	Isi Ember B	Aturan yang dipakai
0	0	2
0	3	9
3	0	2
3	3	17
4	2	5
0	2	9
2	0	Solusi

Contoh Permasalahan 2.

1. Seorang petani akan menyebrangkan seekor kambing, seekor serigala dan sayur mayur dengan sebuah perahu melalui sungai.
2. Perahu hanya bisa memuat petani dan satu penumpang lain.
3. Jika Petani menyebrangkan serigala, sayur akan dimakan kambing
4. Jika Petani menyebrangkan sayur maka kambing akan dimakan serigala

State space identification
Permasalahan ini dapat dilambangkan dengan (Jumlah kambing, jumlah serigala, jumlah sayuran, jumlah petani)

- Keadaan Awal

1. Daerah asal (1,1,1,1)
2. Daerah seberang (0,0,0,0)

- Tujuan

1. Daerah asal (0,0,0,0)
2. Daerah Seberang(1,1,1,1)

Aturan ke	Aturan
1	Kambing menyebrang
2	Sayuran Menyebrang
3	Srigala Menyebrang
4	Kambing Kembali
5	Sayuran Kembali
6	Srigala kembali
7	Boat kembali

Aturan-aturan yang digunakan :
Solusi yang dihasilkan :

Daerah Asal	**Seberang**	**Aturan**
(1,1,1,1)	(0,0,0,0)	1
(0,1,1,0)	(1,0,0,1)	7

(0,1,1,1)	(1,0,0,0)	3
(0,0,1,0)	(1,1,0,1)	4
(1,0,1,1)	(0,1,0,0)	2
(1,0,0,0)	(0,1,1,1)	7
(1,0,0,1)	(0,1,1,0)	1
(0,0,0,0)	(1,1,1,1)	Solusi

Soal :
Carilah Contoh Permasalahan (Misal tentang Game) Jelaskan dari permasalahan tersebut untuk :

a. Mendefinisikan suatu ruang keadaan (state space)
b. Menetapkan satu atau lebih keadaan awal (initial state)
c. Menetapkan satu atau lebih tujuan (goal state)
d. Menetapkan kumpulan aturan

BAB 3
Strategi Pencarian Atau Penelusuran (*Searching*)

Banyak cara yang digunakan untuk membangun sistem yang dapat menyelesaikan masalah - masalah di AI.
Teknik penyelesaian masalah yang dapat dipakai untuk menyelesaikan permasalahan di AI antara lain :

1. Searching (pencarian)
2. Reasoning (penalaran)
3. Planning : memecah masalah menjadi sub - sub masalah satu demi satu untuk kemudian menggabungkan sub sub masalah tersebut menjadi solusi yang lengkap
4. Learning : program komputer yang secara otomatis sanggup belajar dan meningkatkan performance nya melalui pengalaman

Metode Pencarian (Searching) dibagi menjadi 2 yaitu :
1. Blind Search
2. Heuristic Search

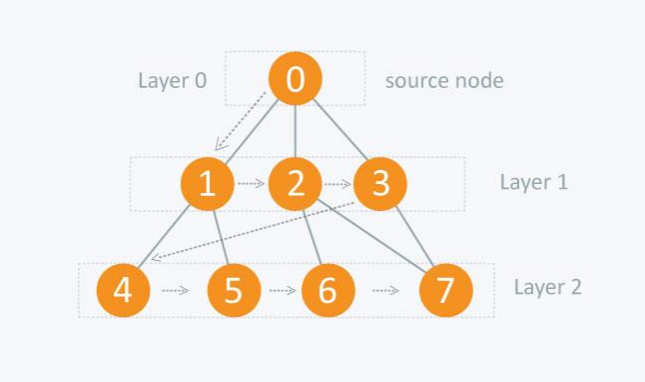
3.1 Metode Blind Search

Metode Blind Search yaitu metode sederhana yang hanya berusaha mencari semua kemungkinan penyelesaian masalah serta tidak ada informasi awal yang bisa digunakan dalam proses pencarian
Algoritma yang digunakan dalam pencarian blind search adalah :

a. Breadth First Search (BFS)
b. Depth First Search (DFS)
c. Uniform Cost Search (UCS)
d. Depth - Limited Search (DLS)
e. Iterative Deepening Search (IDS)
f. Bi - Directional Search (BDS)

3.1.1 Breadth-first Search (BFS)

BFS adalah pendekatan yang paling umum digunakan. BFS adalah algoritma traversing di mana harus mulai melintasi dari node yang dipilih (sumber atau node awal) dan melintasi grafik berlapis-lapis sehingga menjelajahi node tetangga (node yang terhubung langsung ke node sumber). kemudian harus bergerak menuju node tetangga tingkat berikutnya. Seperti namanya BFS, diharuskan untuk melintasi grafik secara luas sebagai berikut:
Pertama bergerak secara horizontal dan kunjungi semua node dari layer saat ini Pindah ke lapisan berikutnya, Perhatikan diagram berikut.



Gambar 3.1 Metode pencarian dengan BFS

Jarak antar node di layer 1 secara komparatif lebih kecil dari jarak antara simpul di layer 2. Untuk itu, di metode ini, harus melintasi semua simpul di layer 1 sebelum pindah ke simpul di layer 2.

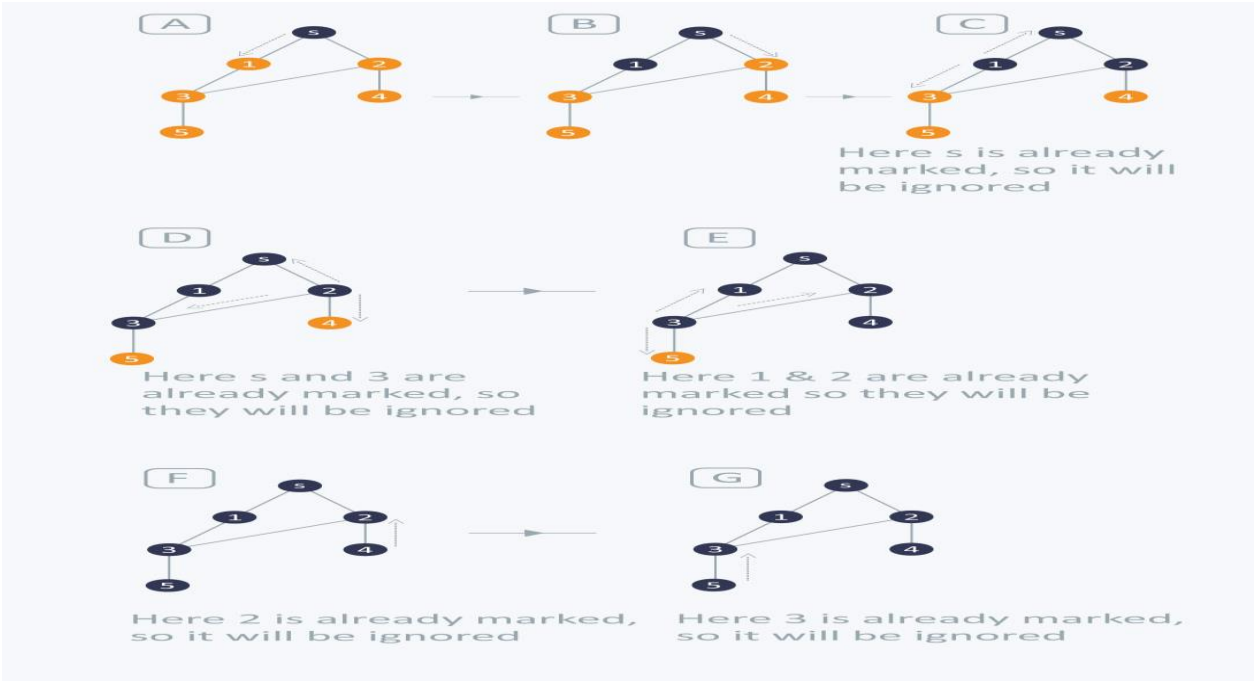
Melintasi node anak

Grafik dapat berisi siklus, yang dapat membawa ke simpul yang sama lagi saat melintasi grafik. Untuk menghindari pemrosesan simpul yang sama lagi, maka array boolean digunakan untuk menandai simpul setelah diproses. Saat mengunjungi simpul di lapisan grafik, simpanlah sedemikian rupa sehingga dapat melintasi simpul anak yang sesuai dalam urutan yang sama.

Gambar yang lalu, mulailah melintasi dari simpul 0 dan kunjungi simpul turunannya yaitu simpul 1, simpul 2, dan simpul 3. Simpan sesuai urutan kunjungannya. Ini akan memungkinkan untuk mengunjungi node anak dari 1 terlebih dahulu (yaitu 4 dan 5), kemudian dari 2 (yaitu 6 dan 7), dan kemudian dari 3 (yaitu 7) dll.

Untuk mempermudah proses ini, gunakan antrian untuk menyimpan simpul dan tandai sebagai 'dikunjungi' sampai semua tetangganya (simpul yang terhubung langsung dengannya) ditandai. Antrian mengikuti metode antrian First In First Out (FIFO), dan oleh karena itu, tetangga dari node akan dikunjungi dalam urutan di mana mereka dimasukkan ke dalam node yaitu node yang dimasukkan terlebih dahulu akan dikunjungi terlebih dahulu, dan seterusnya.

Proses melintasi



Gambar 3.2 Tahapan Metode pencarian dengan BFS

Pelintasan akan dimulai dari node sumber dan mendorong s dalam antrian. s akan diberi tanda sebagai 'dikunjungi'.

Iterasi pertama

Pertama s dikeluarkan dari antrian
Tetangga s yaitu node 1 dan node 2 akan dilalui
Node 1 dan node 2, belum dilalui sebelumnya, akan dilalui. Mereka akan:

- Didorong dalam antrian
- Node 1 dan node 2 diberikan tanda sebagai dikunjungi

Iterasi kedua

Node 1 akan dikeluarkan dari antrian
Tetangga dari node 1 yaitu node s dan node 3 dilalui
Node s diabaikan karena ditandai sebagai 'dikunjungi'
Node 3, yang belum dilalui, maka akan dilalui. Dia:

- Didorong dalam antrian
- Ditandai sebagai dikunjungi

Iterasi ketiga

Node 2 muncul dari antrian
Tetangga dari node 2 yaitu node s, node 3, dan node 4 dilalui
Node 3 dan node s diabaikan karena ditandai sebagai 'dikunjungi'
Node 4, yang belum dilalui, akan dilalui. Dia:

- Didorong dalam antrian
- Ditandai sebagai dikunjungi

Iterasi keempat

Node 3 muncul dari antrian
Tetangga dari node 3 yaitu node1, node 2, dan node 5 dilalui
Node 1 dan node 2 diabaikan karena ditandai sebagai 'dikunjungi'
Node 5, yang belum dilalui sebelumnya, dilalui. Dia:

- Didorong dalam antrian
- Ditandai sebagai dikunjungi

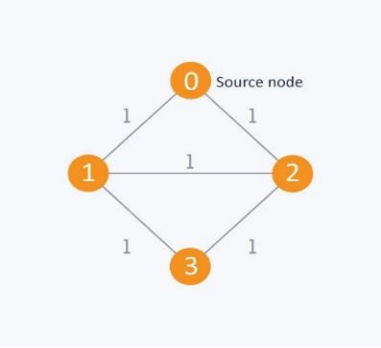
iterasi kelima

node 4 akan muncul dari antrian
Tetangga dari node 4 yaitu node 2 dilalui
Node 2 diabaikan karena sudah ditandai sebagai 'dikunjungi'

iterasi keenam

node 5 muncul dari antrian
Tetangga dari node 5 yaitu node 3 dilalui
Node 3 akan terabaikan karena sudah ditandai sebagai 'dikunjungi'

Antrian kosong dan keluar dari loop. Semua simpul yang telah dilalui dengan menggunakan metode ini. bila sisi yang ada dalam setiap graf mempunyai nilai bobot sama, maka metode ini juga bisa dimanfaatkan dalam mencari jarak minimum antara simpul-simpul didalam sebuah graf.
Contoh :



Gambar 3.3 contoh Metode pencarian dengan BFS

Seperti pada diagram di gambar 3.3 ini, mulailah dari node sumber, untuk mencari jarak antara simpul sumber dan simpul 1. Jika tidak mengikuti algoritma metode ini, maka dapat pergi dari simpul sumber

ke simpul 2 dan kemudian ke simpul 1. Cara berikut akan menghitung nilai jarak diantara simpul sumber dan simpul 1 sebagai simpul 2, untuk jarak minimal sebenarnya yaitu simpul 1. Untuk nilai minimum dari jarak bisa dinilai benar dengan menggunakan algoritma metode ini.

Kompleksitas

Waktu Kompleksitas metode algoritma ini yaitu O*(V + E), di mana V adalah jumlah banyaknya simpul dan E adalah jumlah banyaknya edge.

Aplikasi

1. Cara yang digunakan untuk menentukan level setiap simpul untuk pohon yang diberikan

Seperti yang diketahui di metode ini, untuk melintasi level dengan bijaksana. maka dapat digunakan metode ini, sehingga dapat menentukan level pada setiap simpul.

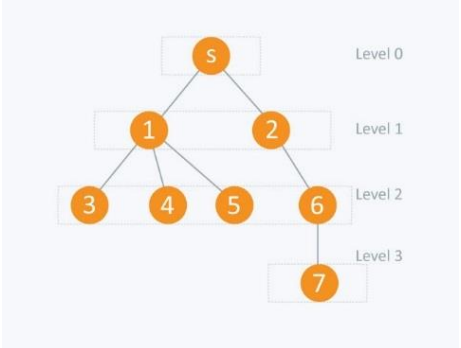
Implementasi :

```
vektor <int> v[10] ;   // Vektor untuk mempertahankan daftar kedekatan yang dijelaskan di atas
   int level[10]; // Untuk menentukan level setiap node
   bool vis[10]; // Tandai simpul jika dikunjungi
   void bfs(int s) {
      queue <int> q;
      q.push(s);
      level[ s ] = 0 ;  // Mengatur level simpul sumber sebagai 0
      vis[ s ] = true;
      while(!q.empty())
      {
         int p = q.front();
         q.pop();
         for(int i = 0;i < v[ p ].size() ; i++)
         {
            if(vis[ v[ p ][ i ] ] == false)
            {
         // Mengatur tingkatan setiap simpul dengan kenaikan level node induk
               level[ v[ p ][ i ] ] = level[ p ]+1;
                q.push(v[ p ][ i ]);
                vis[ v[ p ][ i ] ] = true;
    }
         }
      }
   }
```

Pemrograman ini mirip dengan pemrograman metode BFS hanya mempunyai perbedaan sebagai berikut:

```
tingkat[ v[ p ][ i ] ] = tingkat[ p ]+1;
```

Pemrograman ini, saat mengunjungi setiap simpul, tingkatan simpul tersebut disetel dengan kenaikan level simpul induknya. Ini adalah bagaimana tingkat setiap simpul ditentukan.



Gambar 3.4 Tingkat setiap node ditentukan dalam pencarian dengan BFS

Simpul	level [simpul]
s (source node)	0
1	1
2	1
3	2
4	2
5	2
6	2
7	3

2. 0-1 Metode BFS

Jenis metode BFS digunakan dalam mencari jarak terpendek diantara 2 simpul dalam sebuah graf dengan ketentuan bahwa sisi-sisi dalam graf tersebut mempunyai nilai bobot 0 atau 1. Bila dterapkan metode algoritma ini yang sudah dijelaskan, maka akan didapatkan hasil yang tidak benar dalam jarak optimal antara 2 simpul.

Untuk pendekatan algoritma ini, array boolean tidak digunakan untuk mencari simpul karena kondisi jarak optimal akan diperiksa ketika mengunjungi setiap simpul. Beberapa antrian digunakan untuk menyimpan node. Pada metode 0-1 BFS, jika edge weight = 0, maka node akan terdorong ke depan dequeue. Jika bobot tepi = 1, maka node akan terdorong ke belakang dequeue.

Penerapan

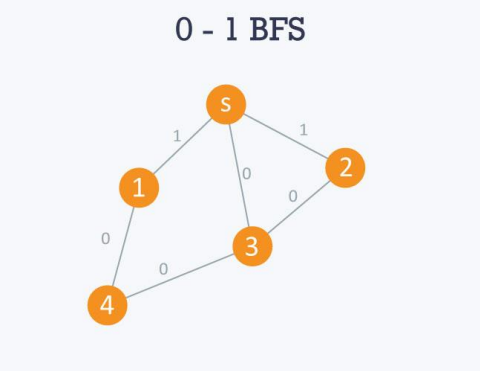
Dalam hal ini, edge[v] [i] merupakan daftar adjacency dalam bentuk pasangan yaitu edge[v][i]. Pertama akan berisi simpul yang terhubung dengan v dan edge[v][i].kedua akan berisi jarak antara v dan tepi[v][i].pertama.

Q adalah antrian ganda. Distance adalah larik dimana, distance[v] akan berisi jarak dari node awal ke node v. Awalnya jarak yang ditentukan dari node sumber ke setiap node adalah tak terhingga.

```
void bfs (int start)
{
        deque <int > Q;     // Antrian berujung ganda
        Q.push_back( start);
        distance[ start ] = 0;
        while( !Q.empty ())
        {
           int v = Q.front( );
           Q.pop_front();
           for( int i = 0 ; i < edges[v].size(); i++)
           {
/* bila jarak tetangga v dari simpul awal lebih besar dari jumlah jarak v dari simpul awal dan bobot tepi
antara v dan tetangganya (jarak antara v dan tetangganya v), maka ubah */
              if(distance[ edges[ v ][ i ].first ] > distance[ v ] + edges[ v ][ i ].second )
           {
              distance[ edges[ v ][ i ].first ] = distance[ v ] + edges[ v ][ i ].second;
           /* Bila nilai bobot tepi antara v dan tetangganya mempunyai nilai 0 maka dorong ke depan
antrian ujung ganda bila tidak dorong ke belakang*/
              if(edges[ v ][ i ].second == 0)
              {
                 Q.push_front( edges[ v ][ i ].first);
              }
              else
              {
                    Q.push_back( edges[ v ][ i ].first);
```

```
            }
        }
    }
}
```

Coba dipahami pemrograman ini dengan grafik berikut:



Gambar 3.5 Grafik 0-1 BFS

Daftar ketetanggaan grafik adalah sebagai berikut:
Di sini 's' dianggap sebagai 0 atau simpul sumber.

0 -> 1 -> 3 -> 2
tepi[0][0].pertama = 1 , tepi[0][0].kedua = 1
tepi[0][1].pertama = 3 , tepi[0][1].kedua = 0
tepi[0][2].pertama = 2 , tepi[0][2].kedua = 1

1 -> 0 -> 4
tepi[1][0].pertama = 0 , tepi[1][0].kedua = 1
tepi[1][1].pertama = 4 , tepi[1][1].kedua = 0

2 -> 0 -> 3
tepi[2][0].pertama = 0 , tepi[2][0].kedua = 0
tepi[2][1].pertama = 3 , tepi[2][1].kedua = 0

3 -> 0 -> 2 -> 4
tepi[3][0].pertama = 0 , tepi[3][0].kedua = 0
tepi[3][2].pertama = 2 , tepi[3][2].kedua = 0
tepi[3][3].pertama = 4 , tepi[3][3].kedua = 0

4 -> 1 -> 3
tepi[4][0].pertama = 1 , tepi[4][0].kedua = 0
tepi[4][1].pertama = 3 , tepi[4][1].kedua = 0

Jika menggunakan algoritma metode BFS, hasilnya akan salah karena akan menunjukkan jarak nilai yang optimal antara simpul s dan simpul 1 dan simpul s dan simpul 2 adalah 1 masing-masing. Ini karena mengunjungi anak-anak simpul s dan menghitung nilai jarak diantara simpul s dan anak-anaknya, adalah 1. Jarak nilai optimal sebenarnya adalah 0 dalam 2 kasus.

Pengolahan

Mulai dari simpul sumber, yaitu 0, ia akan bergerak menuju simpul 1, simpul 2, dan simpul 3. Karena bobot tepi antara simpul 0 dan simpul 1 dan simpul 0 dan simpul 2 berturut-turut yaitu 1, simpul 1 dan

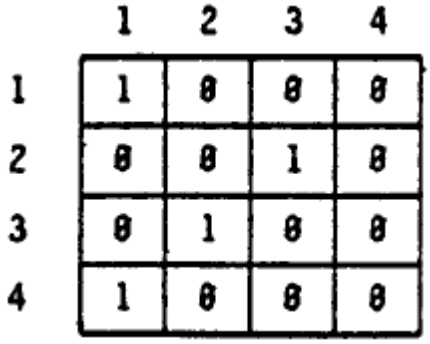
simpul 2 akan didorong ke belakang antrian. Karena nilai bobot tepi antara simpul 0 dan simpul 3 adalah 0, simpul 3 akan didorong ke depan antrian. Jarak akan dipertahankan dalam larik jarak yang sesuai. Simpul 3 kemudian akan muncul dari antrian dan proses yang sama akan diterapkan ke tetangganya, dan seterusnya.

3.1.2 Depth-first Search

Pencarian Depth-First ditandai adanya perluasan node yang paling baru dibuat, atau terdalam, terlebih dahulu. Secara formal, kedalaman sebuah node dalam sebuah pohon didefinisikan sebagai berikut:
Kedalaman simpul awal adalah 0.
Kedalaman simpul lain adalah satu lebih dari kedalaman pendahulunya. Sebagai konsekuensi dari memperluas node terdalam terlebih dahulu, pencarian mengikuti satu jalur melalui ruang keadaan ke bawah dari node awal; hanya jika mencapai keadaan yang tidak memiliki penerus apakah itu mempertimbangkan jalur alternatif. Jalur alternatif secara sistematis memvariasikan yang sebelumnya dicoba, hanya mengubah n langkah terakhir sambil menjaga n sekecil mungkin.
Dalam banyak masalah, tentu saja, pohon ruang keadaan mungkin memiliki kedalaman tak terbatas, atau setidaknya mungkin lebih dalam dari beberapa batas atas yang diketahui pada panjang urutan solusi yang dapat diterima. Untuk mencegah pertimbangan jalur yang terlalu panjang, maksimum sering ditempatkan pada kedalaman node yang akan diperluas, dan setiap node pada kedalaman tersebut diperlakukan seolah-olah tidak memiliki penerus. Perlu dicatat bahwa, bahkan jika batas kedalaman seperti itu digunakan, jalur solusi yang ditemukan belum tentu yang terpendek.
Algoritma berikut menjelaskan pencarian depth-first dengan depth bound

1. Letakkan node awal pada daftar, OPEN, dari node yang tidak digunakan. Jika itu adalah simpul tujuan, solusi telah ditemukan.
2. Jika OPEN kosong, tidak ada solusi.
3. Pindahkan node pertama, n, pada OPEN ke daftar CLOSED dari node yang diperluas.
4. Jika kedalaman simpul n sama dengan kedalaman maksimum, lanjutkan ke (2).
5. Perluas simpul n. Jika tidak memiliki penerus, lanjutkan ke (2).
6. Tempatkan semua penerus node n di awal OPEN.
7. Jika salah satu penerus dari simpul n adalah simpul tujuan, solusi telah ditemukan. Jika tidak, pergi ke (2).

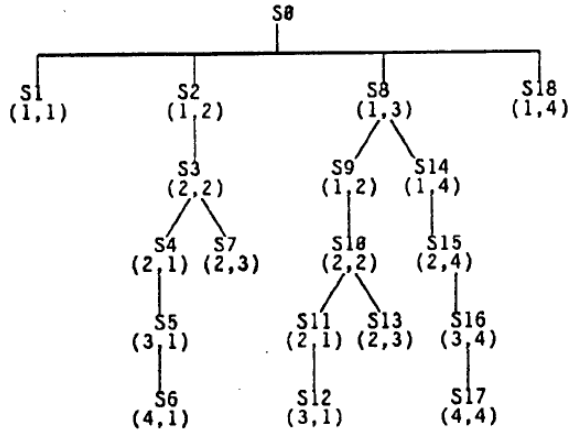
Sebagai contoh, perhatikan masalah sederhana berikut ini: Sebuah pion diperlukan untuk bergerak melalui matriks pada Gambar 3.6 dari atas ke bawah. Pion dapat memasuki matriks di mana saja di baris atas. Dari kotak yang berisi 0, pion harus bergerak ke bawah jika kotak di bawah berisi 0; jika tidak, ia harus bergerak secara horizontal. Dari kotak yang berisi 1, tidak ada gerakan lebih lanjut yang mungkin. Tujuannya adalah untuk mencapai kotak yang berisi nol di baris bawah. Batas kedalaman 5 Diasumsikan.



Gambar 3.6: Contoh masalah untuk pencarian depth-first.

Pohon pencarian yang dihasilkan oleh algoritma depth-first ditunjukkan pada Gambar 3.4. Pada node SO, pion belum memasuki grid. Di node lain, posisinya diberikan sebagai pasangan (nomor baris, nomor kolom). Penomoran node memberikan urutan pemindahannya dari daftar OPEN node yang tidak digunakan. Ketika algoritme berakhir, daftar OPEN berisi Sl7 (node tujuan) dan S18; semua node lain ada di daftar yang diperluas. Solusi yang ditemukan, yaitu satu langkah lebih panjang dari minimum, meminta pion untuk masuk pada (1,3), bergerak satu kotak ke kanan, dan kemudian langsung turun ke

(4,4). Jika tidak ada batasan kedalaman yang digunakan, pohon akan menjadi satu tingkat lebih dalam karena simpul Sl2 memiliki penerus, (4,1). Karena algoritme memperlakukan ruang keadaan sebagai pohon, bukan grafik umum, ia tidak menemukan bahwa simpul yang berbeda S2 dan S9 sebenarnya mewakili keadaan yang sama. Akibatnya, pencarian ke bawah dari S9 menduplikasi pekerjaan yang sudah dilakukan dari S2.



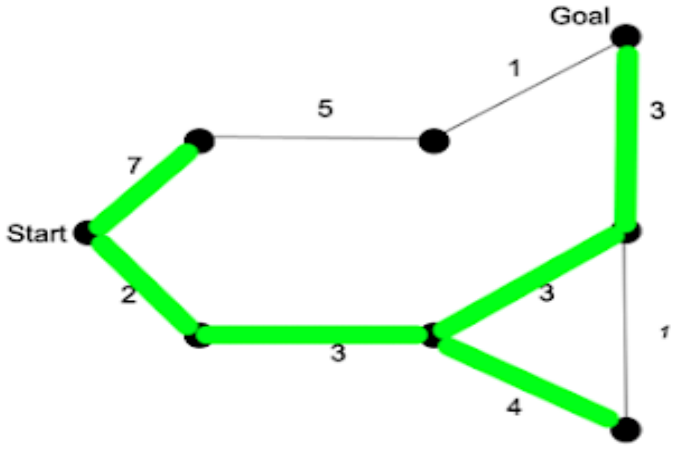
Gambar 3.7: Pohon pencarian untuk Gambar 3.3

3.1.3 Uniform Cost Search

Uniform Cost Search adalah algoritma Seach Tree (graph) yang digunakan untuk menyelesaikan beberapa persoalan . Algoritma ini memulai pencarian dari root node, kemudian dilanjutkan ke node-node selanjutnya. Dimana node tersebut dipilih yang memilki harga (cost) terkecil dari root node. Algoritma ini merupakan modifikasi dari Bread First Search (BFS).

Dalam implementasi algoritma ini , melibatkan semua node yang berhubungan dengan root node, dan meletakannya dalam priority queue untuk mencapai node tujuan. Dimana node – node yang dipilih merupakan node yang berharga terkecil.

Ilustrasi jalannya algoritma Uniform Cost Search dapat digambarkan sebagai berikut :



Gambar 3.8 : Ilustrasi jalannya algoritma Uniform Cost Search

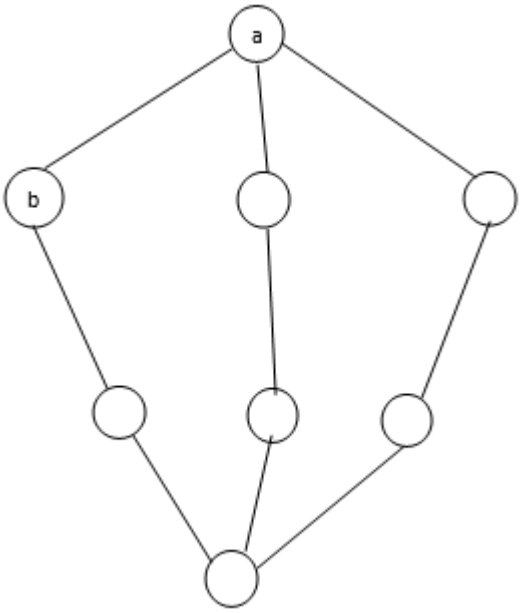
Seperti tampak pada gambar 3.5, initial state terletak pada node start, kemudian untuk mencapai node berikutnya, algoritma ini memilih jalur yang memilki harga terkecil diantara dua node di depannya. Begitu seterusnya, dilakukan pengecekan node yang memilki harga terkecil hingga sampai pada goal state.

UCS adalah algoritma terbaik untuk masalah pencarian, yang tidak melibatkan penggunaan heuristik. Hal ini dapat memecahkan grafik umum untuk biaya yang optimal. UCS kedengarannya pencarian di cabang yang kurang lebih sama dalam biaya.

3.1.4 Depth - Limited Search (DLS)

Depth Limited Search atau disingkat dengan DLS adalah suatu algoritma pencarian dalam menemukan solusi, yaitu sebuah pencarian yang digunakan untuk mengatasi kelemahan dari metode DFS dengan membatasi kedalaman maksimum. Dalam Algoritma ini, digunakan dalam menjalankan dengan membangkitkan pohon pencarian secara dinamis. Pencarian menggunakan metode DFS akan berlanjut sampai kedalaman terakhir dari pohon. Masalah yang muncul dalam metode DFS adalah ketika proses pencarian menemui infinite state space. Hal ini dapat diatasi dengan menginisialisasi batas kedalaman pada tingkatan atau level tertentu dari awal pencarian. Jadi simpul pada tingkat kedalaman akan diperlakukan seolah-olah tidak memiliki penerus. Sebelum menggunakan DLS, terlebih dahulu harus diketahui berapa tingkatan atau level maksimum dari suatu solusi. Kelebihan dari metode DLS yaitu lebih baik dari metode DFS, dimana metode DLS mengatasi kelemahan dari metode DFS. jika depth terlalu kecil, metode DLS juga tidak dapat menemukan solusi yang ada. Ini berarti metode DLS bisa tidak lengkap jika batas kedalaman lebih kecil dari tingkat penyelesaian.

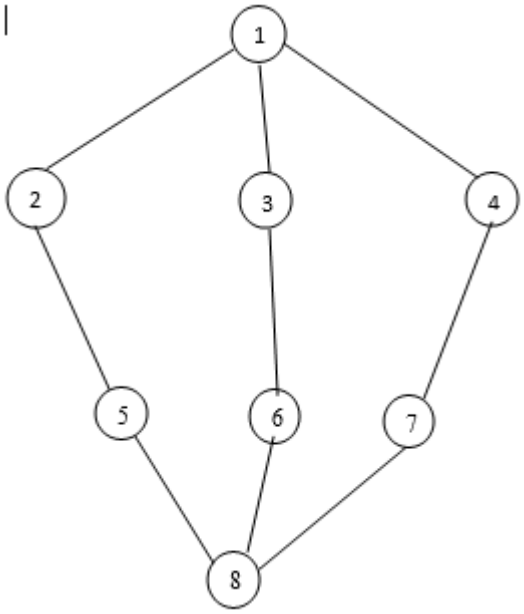
Alur dari metode dalam Dept Limited Search yaitu :



Gambar 3.9 : Langkah-langkah dalam Dept Limited Search

1. Pertama, tentukan node yang akan dicari
2. Kunjungi simpul awal atau simpul a
3. kemudian kunjungi simpul b yang berbatasan pada simpul a yang berada pada batas kedalaman.
4. DLS dimulai dari simpul b ke simpul tetangga. (Mencari simpul dari atas ke bawah lalu ke tetangga dari b dll.)
5. Bila sudah berada pada kedalaman s, maka semua simpul tetangga telah dikunjungi, pencarian di-backtrack ke simpul terakhir.
6. Bila Pencarian telah selesai, maka tidak ada lagi simpul yang belum dikunjungi yang dapat dicapai dari simpul yang dikunjungi dalam kedalaman terbatas.

Sebagai Contoh penerapan algoritma DLS



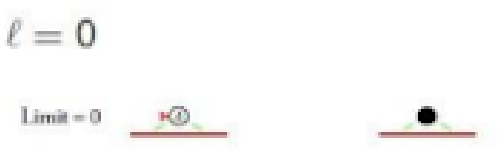
Gambar 3.10 : contoh Langkah-langkah dalam Dept Limited Search

Jika awal simpul bernilai 1 dan batas kedalamannya bernilai 3, urutan yang dikunjunginya adalah simpul 1, 2, 4, 8, 5, 3, 6,.7
dijelaskan dengan jelas bahwa bila kedalaman 3 merupakan simpul 5,6,7 maka untuk sampai pada kedalaman 3 simpul yang akan dilalui untuk yang pertama simpul 1,2,5 maka simpul kedua adalah 1,3,6 dan yang ketiga adalah 1,4 ,7 dan hasil akhir yang ditulis hanya satu kali, maka tidak perlu mengulang angka. Sehingga hasilnya adalah simpul 1,2,3,4,5,6,7.

3.1.5 Iterative Deepening Search (IDS)

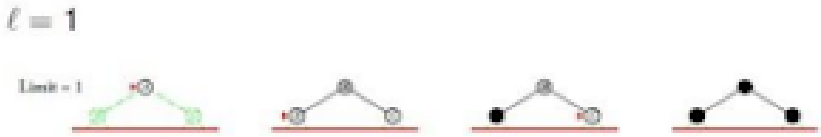
Merupakan metode yang berusaha menggabungkan keuntungan BFS (Complete dan Optimal) dengan keuntungan DFS (Space Complexity yang rendah). Tetapi konsekuensinya adalah Time Complexitynya menjadi tinggi. Pencarian dilakukan secara iteratif (menggunakan penelusuran DFS) dimulai dari batasan level 1. Jika belum ditemukan solusi, maka dilakukan iterasi ke-2 dengan batasan level 2. Demikian seterusnya sampai ditemukan solusi. Jika solusi ditemukan maka tidak diperlukan proses backtracking (penelusuran balik untuk mendapatkan jalur yang dinginkan). Prinsip dari algoritma iterative deepening search adalah untuk melakukan pencarian depth-limited secara bertahap dengan nilai ℓ yang incremental sampai tidak cut off. Proses dibawah ini merupakan langkah dari algoritma IDS ini bekerja.

Proses atau tahapan 1

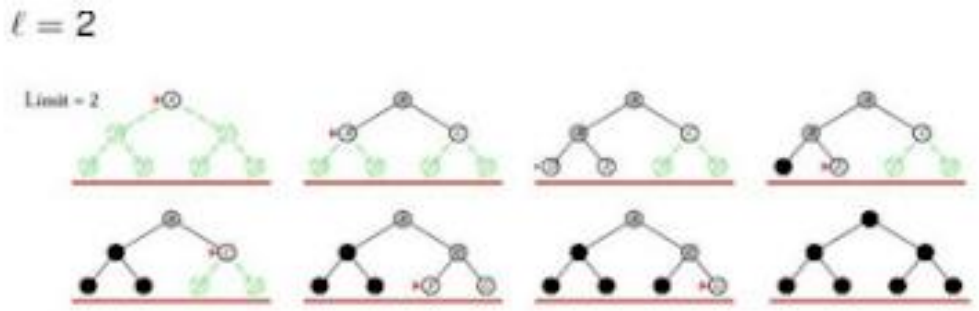


Untuk proses pertama batas kedalaman masih 0 atau kosong sehingga akan terlihat apakah telah ditemukan solusi pada simpul akar. Jika tidak ditemukan maka nilainya akan ditargetkan ke 1

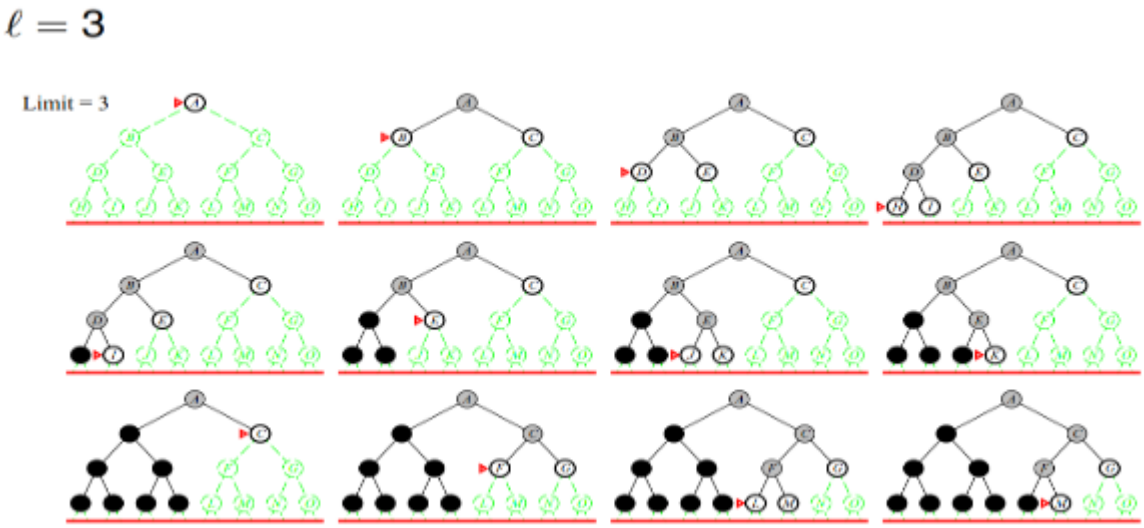
Proses atau tahapan 2



Pada proses kedua batas kedalaman akan bertambah 1 sehingga pencarian solusi secara DFS akan bertambah satu level.
Proses 3



Proses ketiga batas kedalaman sudah menjadi 2 sehingga pencarian akan dilakukan sampai level 2. Jika belum menemukan solusinya, ℓ akan kembali dinaiikan menjadi 3.
Proses 4



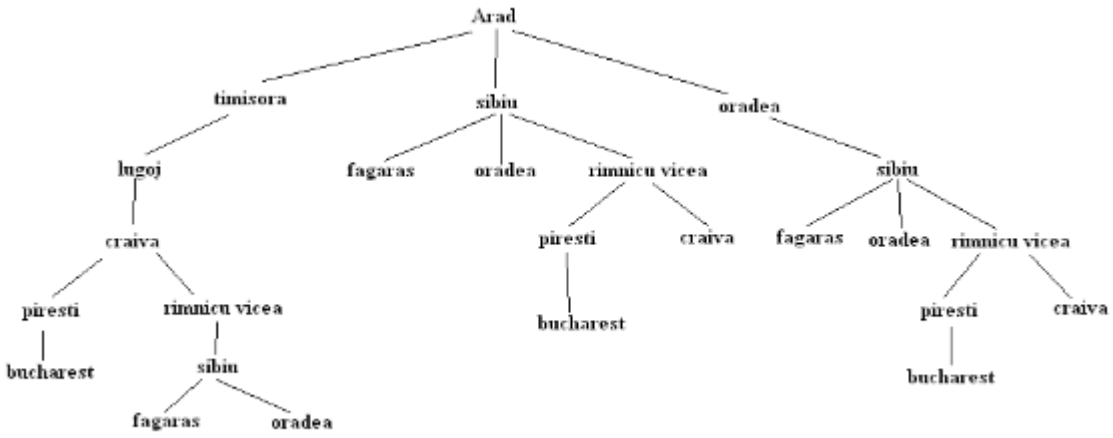
Pada proses ketiga pencarian akan dihentikan pada node M karena itu merupakan goal state

Contoh Kasus :
ROMANIA

Gambar 3.11 : contoh kasus grafik kota

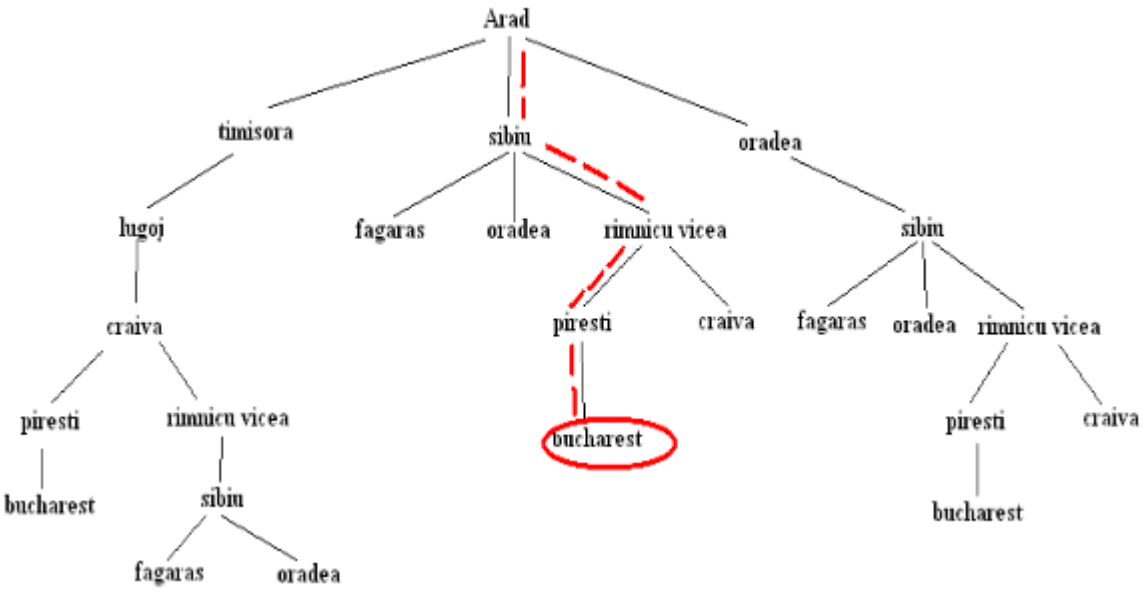
Carilah rute untuk mencapai kota Bucharest dari kota Arad !

Penyelesaian



Gambar 3.12 Bentuk akar dari contoh kasus

Pada contoh kasus kali ini solusi akan ditemukan pada level ke 4 atau pada saat iterasi dari ℓ sama dengan 4.
seperti berikut :



Gambar 3.13 Penyelesaian dengan algoritma IDS

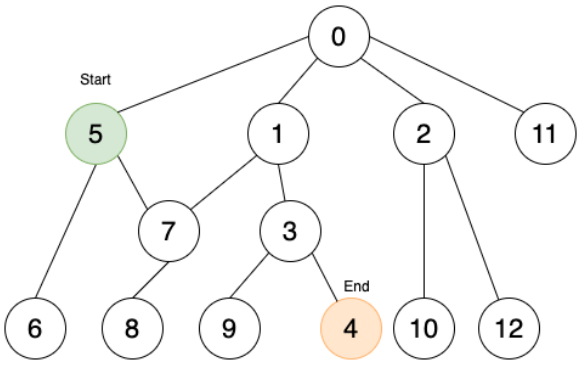
3.1.6 Bi - Directional Search (BDS)

Bidirectional Search adalah Graph Search Algorithm dimana dua graf traversal (BFS) terjadi pada waktu yang sama dan digunakan untuk mencari jarak terpendek antara titik awal tetap dan titik akhir. Ini adalah pendekatan yang lebih cepat, mengurangi waktu yang dibutuhkan untuk melintasi grafik. Bisa juga untuk aplikasi lain.

Ini secara signifikan mengurangi jumlah eksplorasi yang dilakukan. Ini diimplementasikan menggunakan Algoritma Breadth First Search (BFS). (Jika Anda tidak tahu apa itu BFS, lihat artikel ini terlebih dahulu). BFS dijalankan secara bersamaan pada dua simpul - simpul awal dan akhir. Satu pohon BFS sekarang digantikan oleh dua sub pohon, dan pencarian dihentikan ketika dua pohon berpotongan.

Cara kerja

Mari kita coba memahami cara kerja algoritma pencarian Bidirectional melalui contoh berikut.



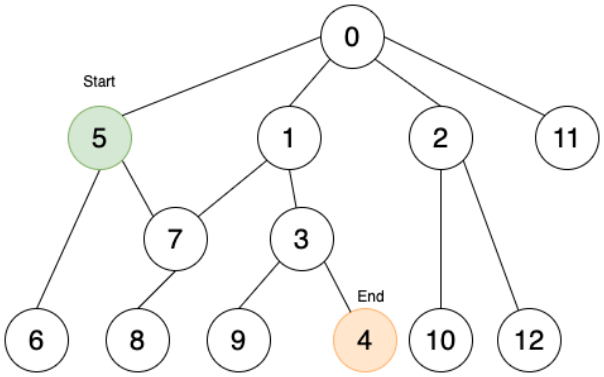
Gambar 3.14 metode algoritma pencarian Bidirectional

Node awal adalah 5 dan node akhir adalah 4.

Tujuan: Untuk menemukan jalur terpendek dari 5 ke 4 menggunakan pencarian dua arah.

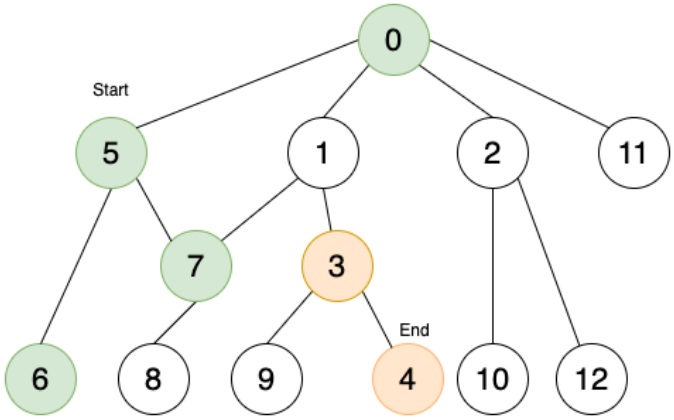
Lakukan BFS dari kedua arah.

1. Mulai bergerak maju dari simpul awal (Hijau) dan mundur dari simpul akhir (Oranye).



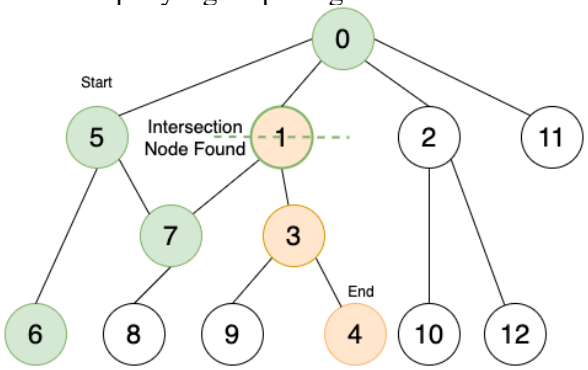
Gambar 3.15 Langkah awal metode algoritma pencarian Bidirectional

2. Mirip dengan BFS, di setiap titik jelajahi level node berikutnya sampai Anda menemukan node yang berpotongan.



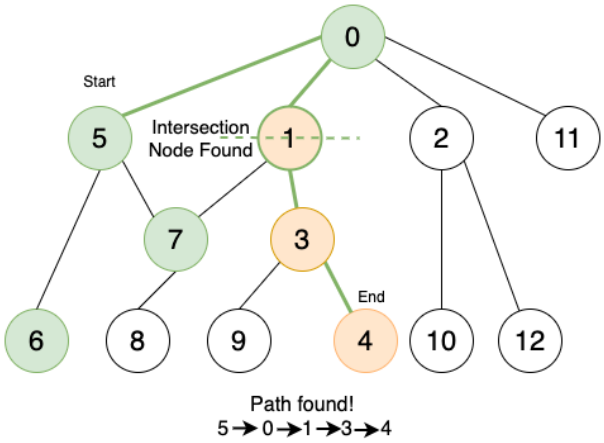
Gambar 3.16 Langkah kedua metode algoritma pencarian Bidirectional

3. Berhentilah menemukan simpul yang berpotongan.



Gambar 3.17 Langkah ketiga metode algoritma pencarian Bidirectional

4. Melacak kembali untuk menemukan jalan



Gambar 3.18 Langkah terakhir metode algoritma pencarian Bidirectional

3.2 Heuristic Search

Heuristik adalah metode yang meningkatkan efisiensi proses pencarian. Ini seperti pemandu wisata. Ada yang baik pada tingkat bahwa mereka mungkin mengabaikan poin dalam arah umum yang menarik; mereka buruk sampai-sampai mereka mengabaikan tempat-tempat menarik bagi individu-individu tertentu. Beberapa heuristik membantu dalam proses pencarian tanpa mengorbankan klaim apa pun atas keseluruhan yang mungkin dimiliki proses sebelumnya. Orang lain kadang-kadang dapat menyebabkan jalan yang sangat baik untuk diabaikan. Dengan mengorbankan keseluruhan itu meningkatkan efisiensi. Heuristik mungkin tidak menemukan solusi terbaik setiap saat tetapi menjamin bahwa mereka menemukan solusi yang baik dalam waktu yang wajar. Ini sangat berguna dalam memecahkan masalah yang sulit dan kompleks, solusi yang akan membutuhkan waktu yang tak terbatas, yaitu jauh lebih lama dari seumur hidup untuk masalah yang tidak diselesaikan di tempat lain.

Untuk menemukan solusi dalam waktu yang tepat daripada solusi lengkap dalam waktu yang tidak terbatas, kami menggunakan heuristik. 'Fungsi heuristik adalah fungsi yang memetakan dari deskripsi keadaan masalah ke ukuran keinginan, biasanya direpresentasikan sebagai angka'. Metode pencarian heuristik menggunakan pengetahuan tentang domain masalah dan memilih operator yang menjanjikan terlebih dahulu. Metode pencarian heuristik ini menggunakan fungsi heuristik untuk mengevaluasi keadaan selanjutnya menuju keadaan tujuan. Untuk menemukan solusi, dengan menggunakan teknik heuristik, seseorang harus melakukan langkah-langkah berikut:

1. Tambahkan domain—informasi spesifik untuk memilih jalur terbaik untuk melanjutkan pencarian.
2. Tentukan fungsi heuristik h(n) yang mengestimasi 'kebaikan' dari sebuah simpul n. Secara khusus, h(n) = perkiraan biaya (atau jarak) dari jalur biaya minimal dari n ke keadaan tujuan.
3. Istilah, heuristik berarti 'melayani untuk membantu penemuan' dan merupakan perkiraan, berdasarkan domain informasi spesifik yang dapat dihitung dari deskripsi keadaan saat ini tentang seberapa dekat kita dengan suatu tujuan.

Menemukan rute dari satu kota ke kota lain adalah contoh dari masalah pencarian di mana urutan pencarian yang berbeda dan penggunaan pengetahuan heuristik mudah dipahami.

1. State space : Kota saat ini tempat pelancong berada.
2. Operator: Jalan yang menghubungkan kota saat ini ke kota lain.
3. Metrik Biaya: Biaya untuk menempuh jalan tertentu antar kota.
4. Informasi heuristik: Pencarian dapat dipandu oleh arah kota tujuan dari kota saat ini, atau kita dapat menggunakan jarak maskapai sebagai perkiraan jarak ke tujuan.

Teknik pencarian heuristik Untuk masalah kompleks, algoritma tradisional, yang disajikan di atas, tidak dapat menemukan solusi dalam beberapa batasan ruang dan waktu praktis. Akibatnya, banyak teknik khusus dikembangkan, menggunakan fungsi heuristik.

- Pencarian heuristik tidak selalu memungkinkan, karena membutuhkan terlalu banyak waktu atau Ruang (memori). Heuristik adalah aturan praktis; mereka tidak menjamin solusi untuk suatu masalah.
- Pencarian Heuristik adalah teknik yang lemah tetapi bisa efektif jika diterapkan dengan benar; itu membutuhkan informasi spesifik domain.

Karakteristik pencarian heuristik

- Heuristik adalah pengetahuan tentang domain, yang membantu pencarian dan penalaran dalam domainnya.
- Pencarian heuristik menggabungkan pengetahuan domain untuk meningkatkan efisiensi dibandingkan pencarian buta.
- Heuristik adalah fungsi yang, ketika diterapkan pada suatu keadaan, mengembalikan nilai sebagai perkiraan nilai keadaan, sehubungan dengan tujuan. Heuristik mungkin (karena alasan) meremehkan atau melebih-lebihkan manfaat suatu keadaan sehubungan dengan tujuan. Heuristik yang meremehkan diinginkan dan disebut dapat diterima.
- Fungsi evaluasi heuristik memperkirakan kemungkinan keadaan tertentu yang mengarah ke keadaan tujuan.
- Fungsi pencarian heuristik memperkirakan biaya dari keadaan saat ini ke tujuan, menganggap fungsi itu efisien.

Contoh: Travelling salesman

Seorang salesman harus mengunjungi daftar kota dan dia harus mengunjungi setiap kota hanya sekali. Ada rute yang berbeda antar kota. Soalnya adalah mencari rute terpendek antar kota sehingga salesman mengunjungi semua kota sekaligus. Misalkan ada N kota, maka solusinya adalah mengambil N! kemungkinan kombinasi untuk menemukan jarak terpendek untuk memutuskan rute yang diperlukan. Ini tidak efisien karena dengan N=10 ada 36.28.800 kemungkinan rute. Ini adalah contoh ledakan kombinatorial. Ada metode yang lebih baik untuk solusi masalah seperti itu: satu disebut cabang dan terikat. Pertama, buat semua jalur lengkap dan temukan jarak jalur lengkap pertama. Jika jalur berikutnya lebih pendek, simpan dan lanjutkan dengan cara ini menghindari jalur saat panjangnya melebihi menyimpan panjang jalur terpendek, meskipun lebih baik dari metode sebelumnya.

Fungsi heuristik digunakan untuk mengevaluasi keadaankeadaan problema individual dan menentukan seberapa jauh hal tersebut dapat digunakan untuk mendapatkan solusi yang diinginkan. Jenis-jenis Heuristic Searching:

1. Generate and Test.
2. HillClimbing.
3. Best First Search.
4. Alpha Beta Prunning, Means-End-Anlysis, Constraint Satisfaction, Simulated Anealing, dll

3.2.1 Generate and Test

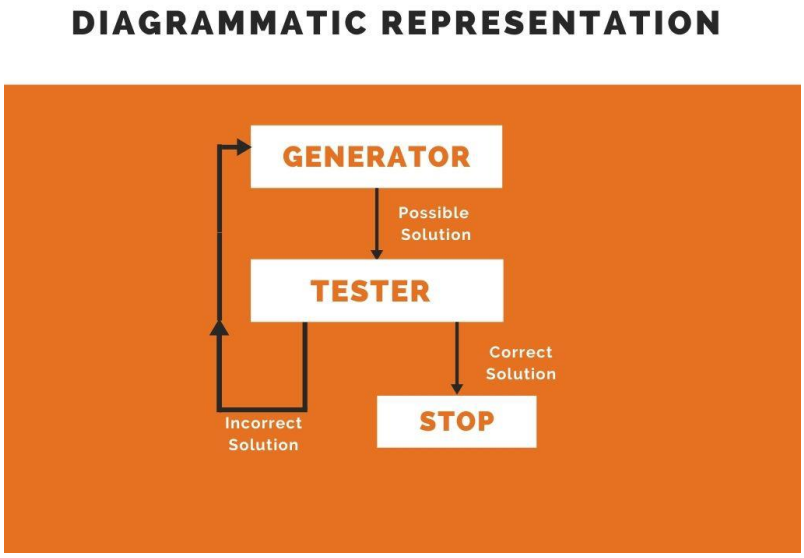
Generate and Test Search adalah teknik pencarian heuristik berbasis Depth First Search dengan Backtracking yang menjamin untuk menemukan solusi jika dilakukan secara sistematis dan ada solusi. Dalam teknik ini, semua solusi dibangkitkan dan diuji untuk solusi terbaik. Ini memastikan bahwa solusi terbaik diperiksa terhadap semua kemungkinan solusi yang dihasilkan.

Disebut juga British Museum Search Algorithm karena seperti mencari pameran secara acak atau menemukan objek di British Museum dengan cara mengembara secara acak.

Evaluasi dilakukan oleh fungsi heuristik karena semua solusi dihasilkan secara sistematis dalam algoritma generate dan test tetapi jika ada beberapa jalur yang paling tidak mungkin membawa kita ke hasil maka mereka tidak dipertimbangkan. Heuristik melakukan ini dengan memberi peringkat semua alternatif dan seringkali efektif dalam melakukannya. Hasilkan dan Uji Sistematis mungkin terbukti tidak efektif saat memecahkan masalah yang kompleks. Tetapi ada teknik untuk meningkatkan dalam kasus yang kompleks juga dengan menggabungkan pencarian menghasilkan dan menguji dengan teknik lain untuk mengurangi ruang pencarian. Misalnya dalam Program Kecerdasan Buatan DENDRAL kami menggunakan dua teknik, yang pertama adalah Teknik Kepuasan Kendala diikuti oleh Generate and Test Procedure untuk bekerja pada ruang pencarian yang dikurangi yaitu menghasilkan hasil yang efektif dengan mengerjakan lebih sedikit daftar yang dihasilkan di bagian paling bawah. Langkah pertama.

Algoritma :

1. Menghasilkan solusi yang mungkin. Misalnya, menghasilkan titik tertentu di ruang masalah atau menghasilkan jalur untuk keadaan awal.
2. Uji untuk melihat apakah ini adalah solusi aktual dengan membandingkan titik yang dipilih atau titik akhir dari jalur yang dipilih dengan himpunan keadaan tujuan yang dapat diterima
3. Jika solusi ditemukan, berhenti. Jika tidak, lanjutkan ke Langkah 1

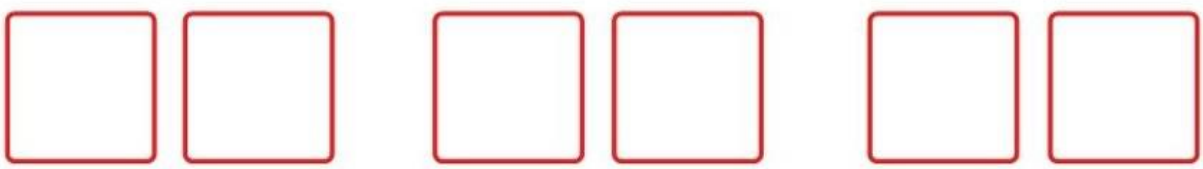


Gambar 3.19 Sifat Generator yang Baik

Properti Generator yang Baik:
Generator yang baik harus memiliki sifat-sifat berikut:

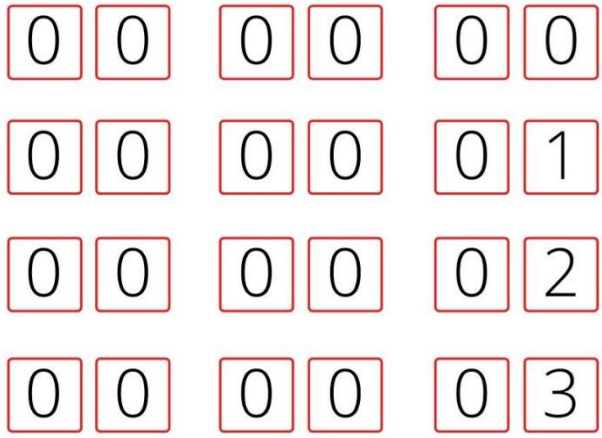
- Lengkap: Generator yang baik harus lengkap yaitu mereka harus menghasilkan semua solusi yang mungkin dan mencakup semua keadaan yang mungkin. Dengan cara ini, kami dapat menjamin algoritme kami untuk konvergen ke solusi yang benar di beberapa titik waktu.
- Non Redundant: Generator yang baik tidak boleh menghasilkan solusi duplikat pada setiap titik waktu karena mengurangi efisiensi algoritma sehingga meningkatkan waktu pencarian dan membuat kompleksitas waktu menjadi eksponensial. Bahkan, sering dikatakan bahwa jika solusi muncul beberapa kali dalam pencarian mendalam-pertama maka lebih baik untuk memodifikasi prosedur untuk melintasi grafik daripada pohon.
- Diinformasikan: Generator yang baik memiliki pengetahuan tentang ruang pencarian yang mereka pertahankan dalam bentuk serangkaian pengetahuan. Ini dapat digunakan untuk mencari seberapa jauh agen dari tujuan, menghitung biaya jalur dan bahkan menemukan cara untuk mencapai tujuan.

Contoh Penerapan Algoritma Generate and Test
Mari kita ambil contoh sederhana untuk memahami pentingnya generator yang baik. Pertimbangkan pin yang terdiri dari tiga angka 2 digit yaitu angkanya berbentuk,



Gambar 3.20 pin yang terdiri dari tiga angka 2 digit

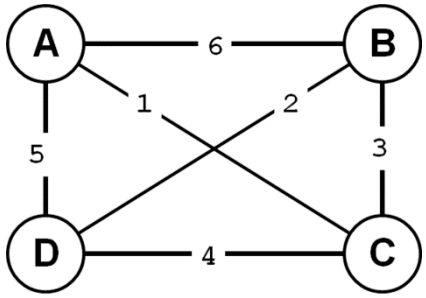
Dalam hal ini, salah satu cara untuk menemukan pin yang dibutuhkan adalah dengan membangkitkan semua solusi secara brute force misalnya,



Gambar 3.21 salah satu cara untuk menemukan pin yang dibutuhkan adalah dengan membangkitkan semua solusi secara brute force

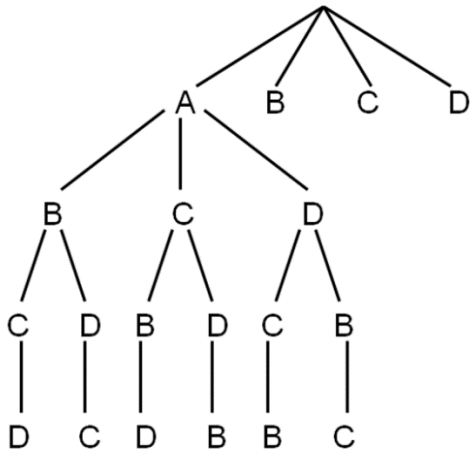
Jumlah total solusi dalam kasus ini adalah (100)3 yang kira-kira 1M. Jadi jika kita tidak menggunakan teknik pencarian informasi apapun maka itu menghasilkan kompleksitas waktu yang eksponensial. Sekarang katakanlah jika kita menghasilkan 5 solusi setiap menit. Maka jumlah total yang dihasilkan dalam 1 jam adalah 5*60=300 dan jumlah total solusi yang akan dihasilkan adalah 1M. Mari kita perhatikan teknik pencarian brute force misalnya pencarian linier yang kompleksitas waktu rata-ratanya adalah N/2. Kemudian rata-rata, jumlah total solusi yang akan dihasilkan adalah sekitar 5 lakh. Menggunakan teknik ini bahkan jika Anda bekerja selama sekitar 24 jam sehari maka Anda juga akan membutuhkan 10 minggu untuk menyelesaikan tugas tersebut.
Sekarang pertimbangkan untuk menggunakan fungsi heuristik di mana kita memiliki pengetahuan domain bahwa setiap bilangan adalah bilangan prima antara 0-99 maka jumlah solusi yang mungkin adalah $(25)^3$ yaitu sekitar 15.000. Sekarang pertimbangkan kasus yang sama bahwa Anda menghasilkan 5 solusi setiap menit dan bekerja selama 24 jam maka Anda dapat menemukan solusi dalam waktu kurang dari 2 hari yang sedang dilakukan dalam 10 minggu dalam kasus pencarian tanpa informasi.
Kita dapat menyimpulkan di sini bahwa jika kita dapat menemukan heuristik yang baik maka kompleksitas waktu dapat dikurangi secara bertahap. Tetapi dalam kasus terburuk, kompleksitas ruang dan waktu akan menjadi eksponensial. Itu semua tergantung pada generator yaitu lebih baik generator lebih sedikit adalah kompleksitas waktu.

Contoh – Travelling Salesman Problem (TSP)
Seorang penjual memiliki daftar kota, yang masing-masing harus dia kunjungi tepat satu kali. Ada jalan langsung antara setiap pasangan kota dalam daftar. Temukan rute yang harus diikuti penjual untuk perjalanan pulang pergi sesingkat mungkin yang dimulai dan diakhiri di salah satu kota.
Wisatawan perlu mengunjungi n kota.
Ketahui jarak antara setiap pasangan kota.
Ingin tahu rute terpendek yang mengunjungi semua kota sekaligus.



Gambar 3.21 Contoh rute Travelling Salesman Problem (TSP)

Alur pencarian dengan Generate and Test



Gambar 3.22 Alur Pencarian

Pencarian dari	Jalur	Panjang Jalur
1	**ABCD**	**19**
2	**ABDC**	**18**
3	**ACBD**	**12**
4	**ACDB**	**13**
5	**ADBC**	**16**
berlanjut		

Terakhir, pilih jalur yang panjangnya kurang.

3.2.2 Hill Climbing

Algoritma pencarian pendakian bukit hanyalah sebuah loop yang terus bergerak ke arah peningkatan nilai. Itu berhenti ketika mencapai "puncak" di mana tidak ada tetangga yang memiliki nilai lebih tinggi. Algoritma ini dianggap sebagai salah satu prosedur paling sederhana untuk mengimplementasikan pencarian heuristik. Mendaki bukit berasal dari ide itu jika Anda mencoba menemukan puncak bukit dan Anda naik ke arah mana pun Anda berada. Heuristik ini menggabungkan keunggulan pencarian depth first dan breadth first menjadi satu metode.

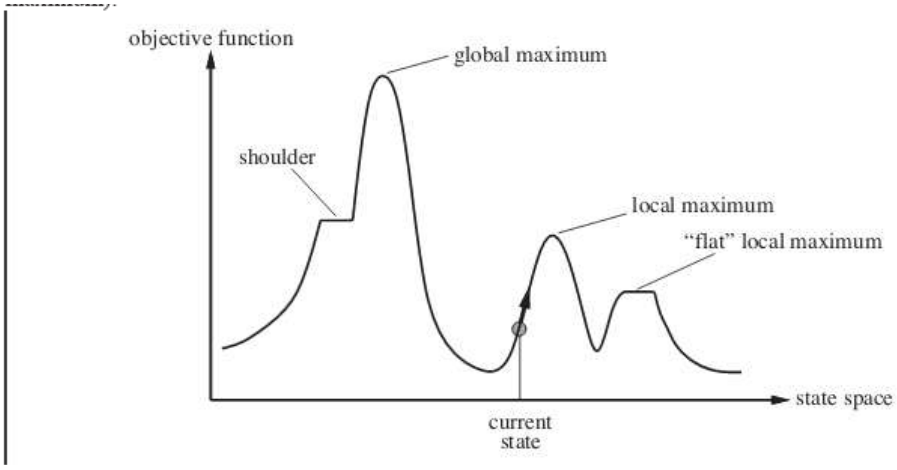
Nama mendaki bukit berasal dari simulasi situasi seseorang mendaki bukit. Orang tersebut akan mencoba untuk bergerak maju ke arah di atas bukit. Pergerakannya berhenti ketika mencapai puncak bukit dan tidak ada puncak yang memiliki nilai fungsi heuristik lebih tinggi dari ini. Pendakian bukit menggunakan pengetahuan tentang medan lokal, memberikan heuristik yang sangat berguna dan efektif untuk

menghilangkan banyak ruang pencarian yang tidak produktif. Ini adalah cabang oleh fungsi evaluasi lokal. Pendakian bukit adalah varian dari generate and test ke arah mana pencarian harus dilanjutkan. Di setiap titik di jalur pencarian, node penerus yang muncul untuk mencapai eksplorasi.
Algoritma:
Langkah 1: Status evaluasi awal. Jika itu adalah keadaan tujuannya maka berhentilah dan kembalikan kesuksesan.
Langkah 2: Jika tidak, lanjutkan dengan keadaan awal dengan mempertimbangkan keadaan saat ini.
Langkah 3: Lanjutkan langkah-4 sampai solusi ditemukan yaitu sampai tidak ada lagi keadaan baru yang tersisa untuk diterapkan pada keadaan saat ini.
Langkah 4:
sebuah. Pilih keadaan yang belum diterapkan ke keadaan saat ini dan terapkan untuk menghasilkan keadaan baru.
b. Prosedur untuk mengevaluasi keadaan baru.
saya. Jika keadaan saat ini adalah keadaan tujuan, maka berhenti dan kembalikan kesuksesan.
ii. Jika lebih baik dari keadaan saat ini, maka buatlah keadaan saat ini dan lanjutkan lebih jauh.
aku aku aku. Jika tidak lebih baik dari keadaan saat ini, maka lanjutkan dalam loop sampai solusi ditemukan.
Langkah 5: Keluar.

Macam – macam metode **Hill Climbing :**

1. Metode Simple Hill climbing, metode ini akan memeriksa simpul dari tetangga satu per satu dan memilih simpul tetangga pertama dengan mengoptimalkan biaya, pada proses simpul berikutnya. Algoritma dari metode ini adalah:
   Langkah 1: periksa dan status awal. Jika simpul adalah tujuannya maka berhenti dan kembali sukses. Jika tidak, buat status awal menjadi status saat ini.
   Langkah 2: hingga keadaan solusi ditemukan atau tidak ada operator baru yang dapat diterapkan ke keadaan saat ini.
   Pilih status yang belum diterapkan ke status saat ini dan terapkan untuk menghasilkan status baru. b) lakukan ini untuk mengembangkan keadaan baru saya. Jika keadaan saat ini adalah keadaan tujuan, maka berhenti dan kembalikan kesuksesan. ii. Jika lebih baik dari keadaan saat ini, maka buat keadaan saat ini dan lanjutkan lebih jauh. I, I, I. Jika tidak lebih baik dari keadaan saat ini, maka lanjutkan dalam loop sampai solusi ditemukan.
   Langkah 3: Keluar.
2. Steepest-Ascent Hill Ascent: hal pertama yang harus dilakukan adalah memeriksa semua simpul tetangga dan kemudian memilih simpul yang paling dekat dengan keadaan solusi sebagai simpul berikutnya.
   Langkah 1: Evaluasi keadaan awal. Jika itu adalah keadaan tujuan, maka keluarlah yang lain, jadikan keadaan saat ini sebagai keadaan awal
   Langkah 2: langkah ini sampai solusi ditemukan atau status saat ini tidak berubah
   i. Biarkan 'target' menjadi keadaan sedemikian rupa sehingga penerus dari keadaan saat ini akan lebih baik darinya;
   ii. untuk setiap operator yang berlaku untuk keadaan saat ini Sebuah. terapkan operator baru dan buat status baru B. mengevaluasi keadaan baru C. jika status ini adalah status tujuan, maka berhentilah jika tidak, bandingkan dengan 'target' D. jika status ini lebih baik dari 'target', tetapkan status ini sebagai 'target' e. jika target lebih baik dari status saat ini, setel status saat ini ke Target
   Langkah 3: Keluar
3. Stochastic hill climbing: metode ini memeriksa secara acak sehingga tidak semua node diperiksa. Metode ini akan memutuskan (berdasarkan peningkatan pada tetangga itu) apakah akan pindah ke tetangga itu atau memeriksa yang lain
   State Space Diagram for Climbing Hills State space diagram adalah representasi grafis dari keadaan perusahaan yang dapat dicapai dengan mencari nilai fungsi tujuan (fungsi yang akan dimaksimalkan). Sumbu X: menunjukkan ruang keadaan yaitu keadaan atau konfigurasi yang mungkin dicapai oleh algoritme kami. Sumbu Y: menunjukkan nilai fungsi tujuan yang sesuai dengan kondisi tertentu.

Solusi terbaik adalah ruang keadaan di mana fungsi tujuan memiliki nilai maksimum (maksimum global).



Gambar 3.22 grafik Stochastic hill climbing

Area Berbeda di bagian Diagram Spasial

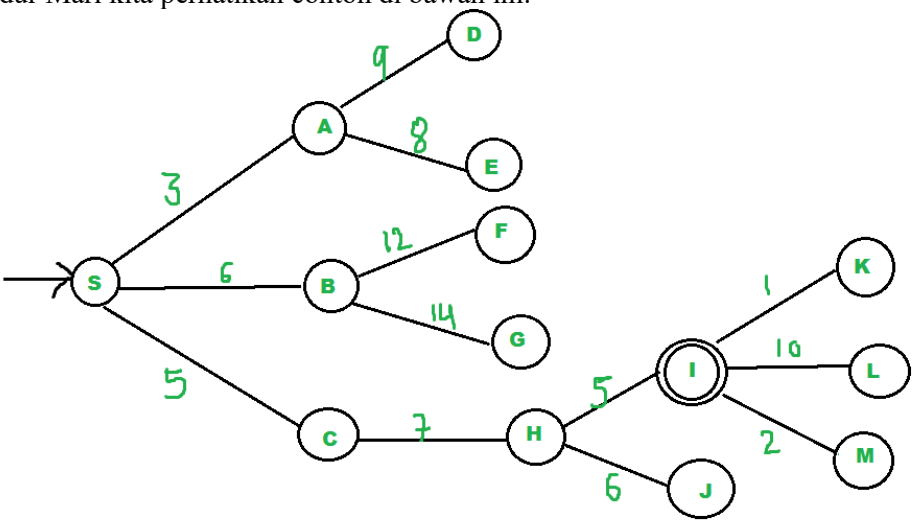
1. Maksimum lokal: Merupakan keadaan yang lebih baik dari keadaan tetangganya tetapi ada keadaan yang lebih baik darinya (maksimum global). Situasi ini lebih baik karena di sini nilai fungsi tujuan lebih tinggi dari tetangganya.
2. Maksimum global: Ini adalah kemungkinan keadaan terbaik dalam diagram ruang keadaan. Hal ini karena dalam keadaan ini, fungsi tujuan memiliki nilai tertinggi.
3. Plateua/flat local maximum: Ini adalah area datar dari ruang keadaan dimana negara-negara tetangga memiliki nilai yang sama.
4. Ridge: Daerah yang lebih tinggi dari tetangganya tetapi memiliki kemiringan. Ini adalah jenis lokal kustom maksimum.
5. Status saat ini: Diagram area keadaan tempat kita berada saat ini selama pencarian.
6. Bahu: Ini adalah dataran tinggi yang memiliki tepi menanjak

Permasalahan di berbagai area dalam mendaki bukit Pendakian bukit tidak dapat mencapai kondisi optimal/terbaik (global maximum) jika masuk ke salah satu area berikut: 1. Local maximum : Pada local maximum semua negara tetangga memiliki nilai yang lebih buruk dari saat ini negara. Karena mendaki bukit menggunakan pendekatan serakah, itu tidak akan mengubah keadaan menjadi lebih buruk dan berakhir dengan sendirinya. Proses akan berakhir meskipun solusi yang lebih baik mungkin ada. Untuk mengatasi masalah lokal secara maksimal: teknik backtracking. Pertahankan daftar negara yang dikunjungi. Jika status pencarian tidak diinginkan, ia dapat kembali ke konfigurasi sebelumnya dan menjelajahi jalur baru. 2. Dataran Tinggi: Di dataran tinggi semua tetangga memiliki nilai yang sama. Oleh karena itu, tidak mungkin untuk memilih arah yang terbaik. Untuk mengatasi dataran tinggi: lompatan besar. Pilih secara acak keadaan yang jauh dari keadaan saat ini. Mengarahkannya adalah bahwa kita akan mendarat di daerah non-dataran tinggi 3. Punggungan: Setiap titik di punggungan dapat terlihat seperti puncak karena pergerakan ke segala arah yang mungkin adalah ke bawah. Oleh karena itu algoritma berhenti ketika mencapai keadaan ini. Untuk mengatasi Ridge : Dalam halangan seperti itu, gunakan dua atau lebih aturan sebelum pengujian. Ini menyiratkan bergerak ke beberapa arah sekaligus.

3.2.3 Best First Search

Dalam BFS dan DFS, ketika kita berada di sebuah node, kita dapat menganggap salah satu dari yang berdekatan sebagai node berikutnya. Jadi baik BFS dan DFS secara membabi buta mengeksplorasi jalur tanpa mempertimbangkan fungsi biaya apa pun. Ide Pencarian Pertama Terbaik adalah menggunakan fungsi evaluasi untuk memutuskan tetangga mana yang paling menjanjikan dan kemudian mengeksplorasi. Pencarian Pertama Terbaik termasuk dalam kategori Pencarian Heuristik atau Pencarian

Informasi. Kami menggunakan antrian prioritas untuk menyimpan biaya node. Jadi implementasinya adalah variasi dari BFS, kita hanya perlu mengubah Queue menjadi PriorityQueue.
Algorithm:
Pencarian Terbaik Pertama (Grafik g, Node mulai)
1) Buat PriorityQueue kosong Prioritas Antrian pq;
2) Masukkan "mulai" di pq. pq.insert(mulai)
3) Sampai PriorityQueue kosong
u = PriorityQueue.
DeleteMin
Jika kamu adalah tujuannya keluar Lain Foreach tetangga v dari u Jika v "Belum dikunjungi" Tandai v "Dikunjungi" pq.insert(v) Tandai v "Diperiksa"
Akhiri prosedur Mari kita perhatikan contoh di bawah ini.



Gambar 3.23 Contoh grafik BFS

mulai dari sumber "S" dan mencari tujuan "I" menggunakan biaya yang diberikan dan Terbaik Pencarian pertama. pq awalnya berisi S Kami menghapus s dari dan memproses yang belum dikunjungi tetangga dari S ke pq. pq sekarang berisi {A, C, B} (C diletakkan sebelum B karena C memiliki biaya lebih rendah) Kami menghapus A dari pq dan memproses yang belum dikunjungi tetangga A ke pq. pq sekarang berisi {C, B, E, D}
menghapus C dari pq dan memproses yang belum dikunjungi tetangga dari C ke pq. pq sekarang berisi {B, H, E, D} Kami menghapus B dari pq dan memproses yang belum dikunjungi tetangga B ke pq. pq sekarang berisi {H, E, D, F, G} Kami menghapus H dari pq. Sejak tujuan kita "Saya" adalah tetangga dari H, kami kembali.
Analisis :

- Kompleksitas waktu kasus terburuk untuk Pencarian Pertama Terbaik adalah O(n * Log n) di mana n adalah angka dari node. Dalam kasus terburuk, kita mungkin harus mengunjungi semua node sebelum mencapai tujuan. Perhatikan bahwa antrian prioritas diimplementasikan menggunakan Min(atau Max) Heap, dan operasi penyisipan dan penghapusan membutuhkan waktu O(log n).
- Kinerja algoritma tergantung pada seberapa baik biaya atau fungsi evaluasi dirancang.

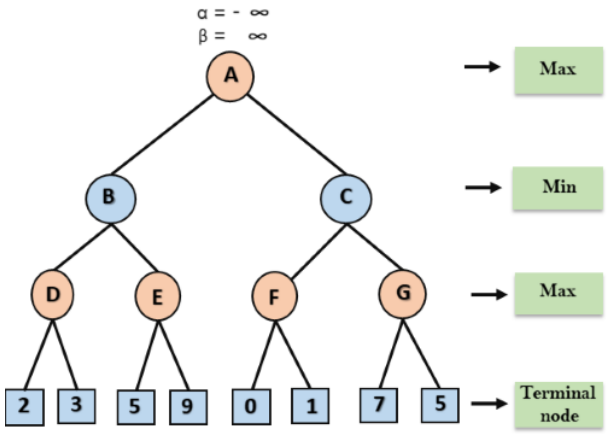
3.2.4 Alpha Beta Prunning

Pemangkasan alfa-beta adalah versi modifikasi dari algoritma minimax. Ini adalah teknik optimasi untuk algoritma minimax.
Seperti yang telah kita lihat dalam algoritme pencarian minimax bahwa jumlah status permainan yang harus diperiksa adalah eksponensial di kedalaman pohon. Karena kita tidak bisa menghilangkan eksponennya, tapi kita bisa memotongnya menjadi setengahnya. Oleh karena itu ada teknik yang tanpa memeriksa setiap simpul dari pohon permainan kita dapat menghitung keputusan minimax yang benar,

dan teknik ini disebut pemangkasan. Ini melibatkan dua parameter ambang Alpha dan beta untuk ekspansi di masa depan, sehingga disebut pemangkasan alfa-beta. Ini juga disebut sebagai Algoritma Alpha-Beta. Pemangkasan alfa-beta dapat diterapkan pada setiap kedalaman pohon, dan terkadang tidak hanya memangkas daun pohon tetapi juga seluruh sub-pohon.
Dua parameter dapat didefinisikan sebagai:

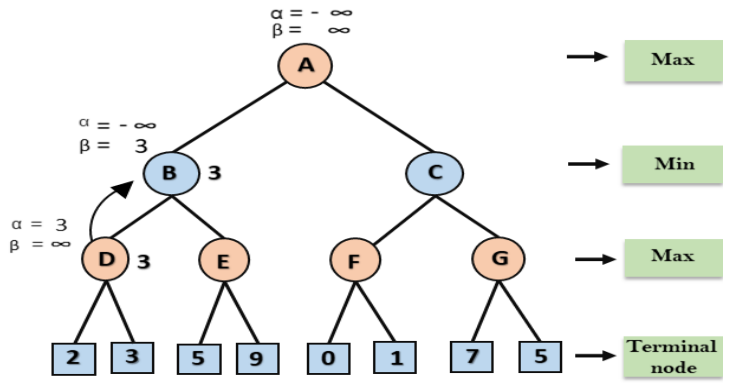
- Alpha: Pilihan terbaik (nilai tertinggi) yang kami temukan sejauh ini di titik mana pun di sepanjang jalur Maximizer. Nilai awal alpha adalah -∞.
- Beta: Pilihan terbaik (nilai terendah) yang kami temukan sejauh ini di titik mana pun di sepanjang jalur Minimizer. Nilai awal beta adalah +∞.

Pemangkasan Alpha-beta ke algoritma minimax standar mengembalikan langkah yang sama seperti algoritma standar, tetapi menghapus semua node yang tidak benar-benar mempengaruhi keputusan akhir tetapi membuat algoritma lambat. Oleh karena itu dengan memangkas simpul-simpul ini, itu membuat algoritma menjadi cepat.
Kondisi utama yang diperlukan untuk pemangkasan alfa-beta adalah:
α>=β
Poin-poin penting tentang pemangkasan alfa-beta:
Pemain Max hanya akan memperbarui nilai alpha.
Pemain Min hanya akan memperbarui nilai beta.
Saat menelusuri kembali pohon, nilai simpul akan diteruskan ke simpul atas alih-alih nilai alfa dan beta.
hanya akan meneruskan nilai alfa, beta ke node anak.
Cara Kerja Alpha-Beta Pruning:
Mari kita ambil contoh pohon pencarian dua pemain untuk memahami cara kerja Alpha-Beta Pruning
Langkah 1:
Pada langkah pertama, Max player akan memulai langkah pertama dari node A dimana = -∞ dan = +∞, nilai alpha dan beta ini diturunkan ke node B dimana lagi = -∞ dan = + , dan Node B memberikan nilai yang sama ke anaknya D.



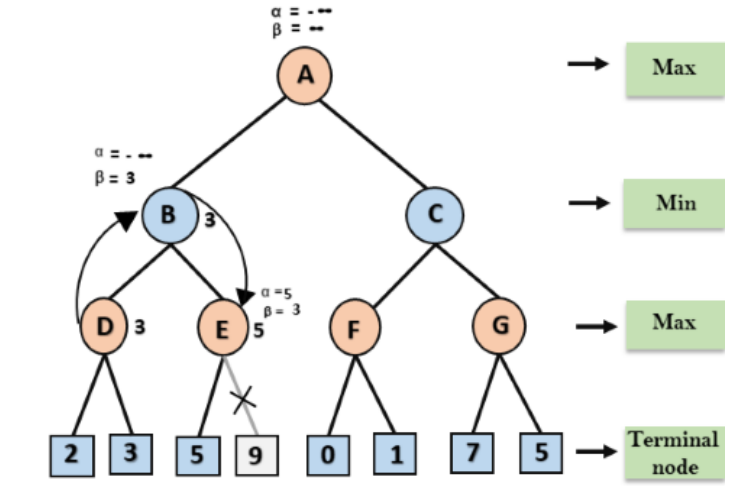
Gambar 3.24 contoh pohon pencarian dua pemain untuk memahami cara kerja Alpha-Beta Pruning

Langkah 2:
Pada Node D, nilai α akan dihitung sebagai gilirannya untuk Max. Nilai α dibandingkan dengan pertama 2 dan kemudian 3, dan maks (2, 3) = 3 akan menjadi nilai pada simpul D dan nilai simpul juga 3.
Langkah 3:
Sekarang algoritma mundur ke node B, di mana nilai β akan berubah karena ini adalah giliran Min, Sekarang β = +∞, akan dibandingkan dengan nilai node berikutnya yang tersedia, yaitu min (∞, 3) = 3, maka pada simpul B sekarang α = -∞, dan β = 3.



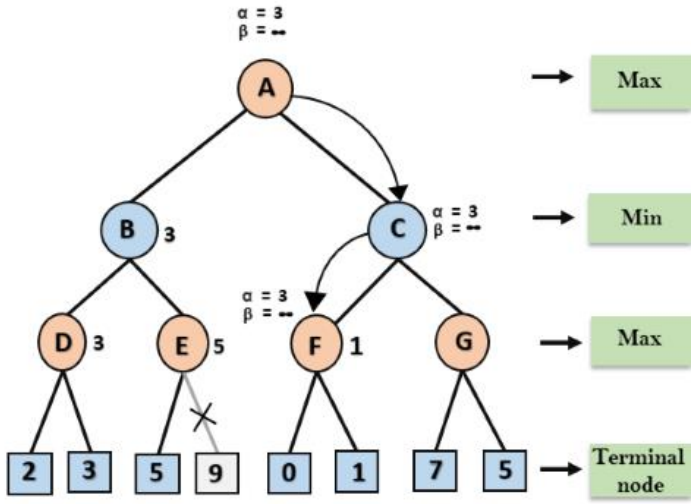
Gambar 3.25 Langkah 3 pohon pencarian dua pemain untuk memahami cara kerja Alpha-Beta Pruning

Pada langkah selanjutnya, algoritma melintasi penerus berikutnya dari Node B yaitu node E, dan nilai α = -∞, dan β = 3 juga akan dilewatkan.
Langkah 4:
Pada node E, Max akan mengambil gilirannya, dan nilai alpha akan berubah. Nilai alpha saat ini akan dibandingkan dengan 5, sehingga max (-∞, 5) α = 5, maka pada node E = 5 dan β = 3, dimana α >=β, maka penerus kanan E akan dipangkas, dan algoritma tidak akan melewatinya, dan nilai pada node E akan menjadi 5.



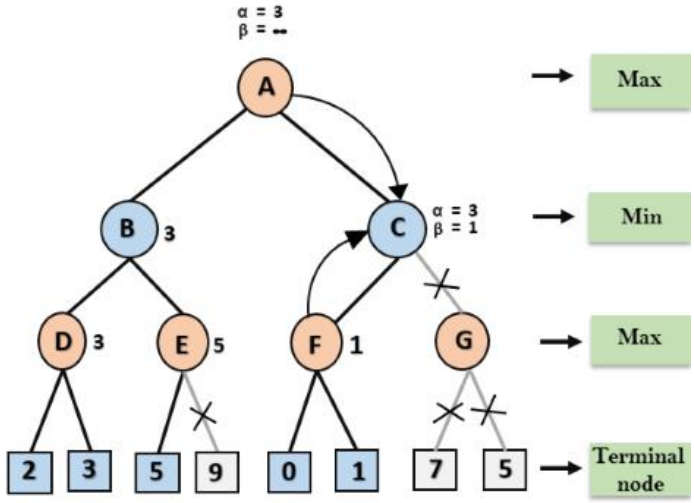
Gambar 3.26 Langkah pohon pencarian dua pemain untuk memahami cara kerja Alpha-Beta Pruning

Langkah 5: Pada langkah selanjutnya, algoritma kembali melakukan backtrack pada pohon, dari node B ke node A. Pada node A, nilai alpha akan diubah nilai maksimum yang tersedia adalah 3 sebagai max (-∞, 3) α = 3, dan β = +∞, kedua nilai ini sekarang diteruskan ke penerus kanan A yaitu Node C.
Pada simpul C, =3 dan β = +∞, dan nilai yang sama akan diteruskan ke simpul F.
Langkah 6: Pada simpul F, kembali nilai akan dibandingkan dengan anak kiri yaitu 0, dan max(3,0)= 3, kemudian dibandingkan dengan anak kanan yaitu 1, dan max(3,1)= 3 masih tetap 3, tetapi nilai simpul F akan menjadi 1.



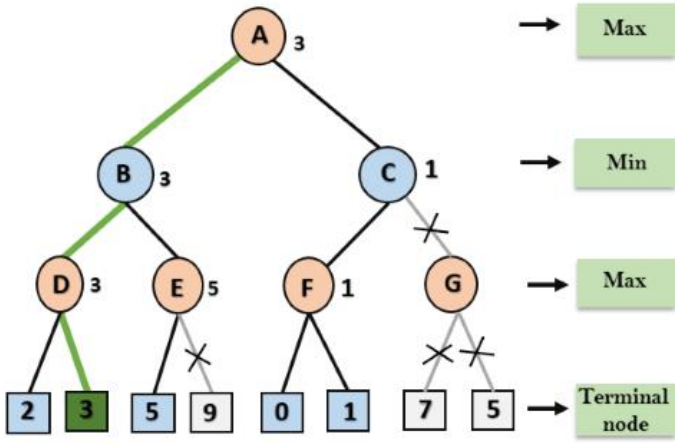
Gambar 3.27 Langkah 6 pohon pencarian dua pemain untuk memahami cara kerja Alpha-Beta Pruning

Langkah 7: Node F mengembalikan nilai node 1 ke node C, pada C α = 3 dan β = +∞, di sini nilai beta akan diubah, itu akan dibandingkan dengan 1 jadi min (∞, 1) β = 1. Sekarang pada C, α =3 dan = 1, dan lagi memenuhi kondisi α >=β, sehingga anak C berikutnya yaitu G akan dipangkas, dan algoritma tidak akan menghitung seluruh sub-pohon G.



Gambar 3.28 Langkah 7 pohon pencarian dua pemain untuk memahami cara kerja Alpha-Beta Pruning

Langkah 8: C sekarang mengembalikan nilai 1 ke A di sini nilai terbaik untuk A adalah maks (3, 1) = 3. Berikut adalah pohon permainan terakhir yang menunjukkan node yang dihitung dan node yang tidak pernah dihitung. Oleh karena itu nilai optimal untuk pemaksimal adalah 3 untuk contoh ini.



Gambar 3.29 Langkah 8 pohon pencarian dua pemain untuk memahami cara kerja Alpha-Beta Pruning

Soal :

Buatlah Skema Peta Provinsi Jawa Timur, dari peta tersebut hubungkan kota dengan kota yang lain kemudian beri jarak berapa kilometer antar kota tersebut.

1. Gambar peta jawa timur yang sudah ada jarak antar kota
2. Start dan tujuan (dari kota mana dan tujuan kekota mana)
3. Gambarkan dalam bentuk graf

Gunakan metode pencarian Blind Search dan Heuristic Search untuk pencari jalur terpendek dari dua kota yang ditempuh.

BAB 4
Representasi Pengetahuan

4.1 Representasi Pengetahuan

Untuk tujuan memecahkan masalah kompleks yang dihadapi dalam AI, kita membutuhkan sejumlah besar pengetahuan dan beberapa mekanisme untuk memanipulasi pengetahuan itu untuk menciptakan solusi bagi masalah baru. Berbagai teknik dalam merepresentasikan pengetahuan (fakta) telah dieksploitasi dalam program AI. Dalam semua variasi representasi pengetahuan, kita berurusan dengan dua jenis entitas.

A. Fakta: Kebenaran di beberapa dunia yang relevan. Ini adalah hal-hal yang ingin kami wakili.

B. Representasi fakta dalam beberapa formalisme yang dipilih. ini adalah hal-hal yang sebenarnya dapat kita manipulasi.

Salah satu cara untuk memikirkan penataan entitas ini adalah pada dua tingkat:

(a) tingkat pengetahuan, di mana fakta dijelaskan, dan

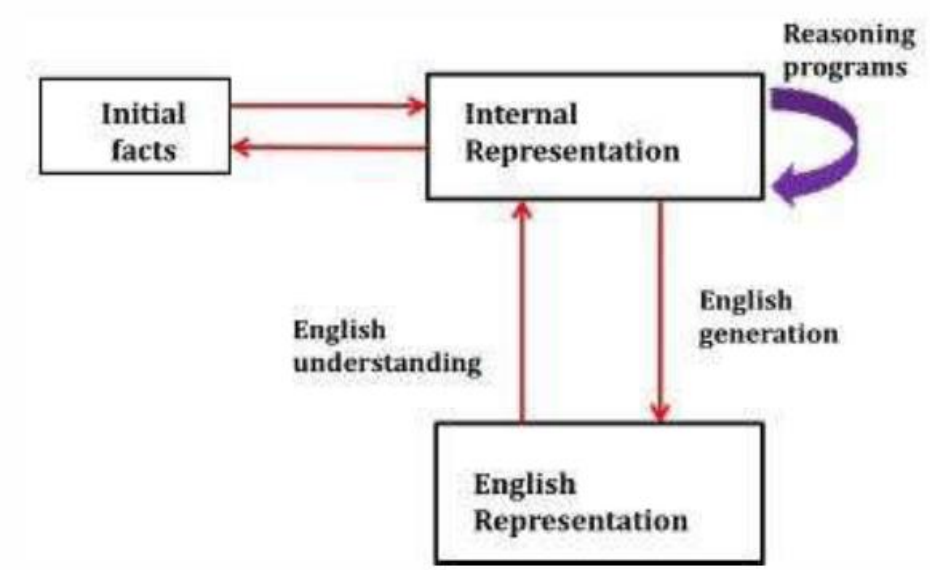
(b) tingkat simbol, di mana representasi objek pada tingkat pengetahuan didefinisikan dalam bentuk simbol yang dapat dimanipulasi oleh program.

Fakta dan representasi dihubungkan dengan pemetaan dua arah. Tautan ini disebut pemetaan representasi. Pemetaan representasi maju memetakan dari fakta ke representasi. Pemetaan representasi ke belakang berjalan sebaliknya, dari representasi ke fakta.

Salah satu representasi umum adalah kalimat bahasa alami (khususnya bahasa Inggris). Terlepas dari representasi fakta yang kami gunakan dalam suatu program , kami mungkin juga perlu memperhatikan representasi bahasa Inggris dari fakta-fakta tersebut untuk memfasilitasi masuk dan keluarnya informasi dari sistem. Kami membutuhkan fungsi pemetaan dari kalimat bahasa Inggris ke representasi yang sebenarnya digunakan dan dari itu kembali ke kalimat.

Representasi dan Pemetaan

- Untuk memecahkan masalah kompleks yang dihadapi dalam kecerdasan buatan, seseorang membutuhkan sejumlah besar pengetahuan dan beberapa mekanisme untuk memanipulasi pengetahuan itu untuk menciptakan solusi.
- Pengetahuan dan Representasi adalah dua entitas yang berbeda. Mereka memainkan peran sentral tetapi dapat dibedakan dalam sistem cerdas.
- Pengetahuan adalah deskripsi dunia. Ini menentukan kompetensi sistem dengan apa yang diketahuinya.
- Selain itu, Representasi adalah cara pengetahuan dikodekan. Ini mendefinisikan kinerja sistem dalam melakukan sesuatu.
- Berbagai jenis pengetahuan membutuhkan jenis representasi yang berbeda.



Gambar 4.1 Pemetaan antara Fakta dan Representasi

Representasi pengetahuan adalah suatu cara untuk mempresentasikan apa yang diperoleh dalam suatu skema/diagram tertentu sehingga hubungan antara pengetahuan dengan pengetahuan lain dapat diketahui dan dapat digunakan untuk menguji kebenaran penalarannya.
Secara teknis, representasi pengetahuan dibagi menjadi tujuh kelompok:
1. Representasi Logis
2. Daftar
3. Pohon
4. Jaringan Semantik
5. Bingkai
6. Naskah (Script)
7. Aturan Produksi (Production Rules)

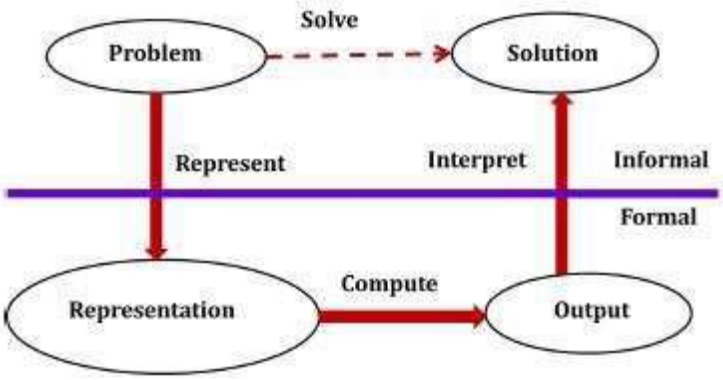
Pengetahuan dikategorikan menjadi dua jenis utama:
1. Tacit sesuai dengan "informal" atau "implisit"
   - Ada dalam diri manusia;
   - Itu diwujudkan.
   - Sulit untuk diartikulasikan secara formal.
   - Sulit untuk berkomunikasi atau berbagi.
   - Selain itu, Sulit untuk dicuri atau disalin.
   - Diambil dari pengalaman, tindakan, wawasan subjektif
2. Jenis pengetahuan formal eksplisit, Eksplisit
   - Pengetahuan Eksplisit
   - Ada di luar manusia;
   - Itu tertanam.
   - Dapat diartikulasikan secara formal.
   - Juga, Dapat dibagikan, disalin, diproses, dan disimpan.
   - Jadi, Mudah dicuri atau disalin
   - Diambil dari artefak dari beberapa jenis sebagai prinsip, prosedur, proses, konsep.

Berbagai teknik dalam merepresentasikan pengetahuan telah dieksploitasi dalam pemrograman kecerdasan buatan.
Terdapat dua macam entitas yang berbeda, yang sedang kita hadapi.
1. Fakta: Kebenaran di beberapa dunia yang relevan. Hal-hal yang ingin kami wakili.

2. Juga, Representasi fakta dalam beberapa formalisme yang dipilih. Hal-hal yang sebenarnya bisa kita lakukan untuk memanipulasi. Entitas ini terstruktur pada dua tingkat:
    1. Tingkat pengetahuan, di mana fakta dijelaskan.
    2. Selain itu, tingkat simbol, di mana representasi objek didefinisikan dalam hal simbol yang dapat dimanipulasi oleh program

Kerangka Representasi Pengetahuan

- Komputer memerlukan deskripsi masalah yang terdefinisi dengan baik untuk memproses dan memberikan solusi yang dapat diterima dengan baik.
- Selain itu, Untuk mengumpulkan bagian-bagian pengetahuan, pertama-tama kita perlu merumuskan deskripsi dalam bahasa lisan kita dan kemudian merepresentasikannya dalam bahasa formal sehingga komputer dapat memahaminya.
- Selain itu, komputer kemudian dapat menggunakan algoritme untuk menghitung jawaban. Jadi, Proses ini digambarkan sebagai,



Gambar 4.2 Kerangka Representasi Pengetahuan

Langkah-langkahnya adalah:

- Formalisme informal dari masalah terjadi terlebih dahulu.
- Kemudian direpresentasikan secara formal dan komputer menghasilkan output.
- Output ini kemudian dapat direpresentasikan dalam solusi yang dijelaskan secara informal yang dipahami atau diperiksa oleh pengguna untuk konsistensi.

Pemecahan masalah membutuhkan,

- Representasi pengetahuan formal, dan
- Selain itu, Konversi pengetahuan informal menjadi pengetahuan formal yaitu konversi pengetahuan implisit menjadi pengetahuan eksplisit.

Pemetaan antara Fakta dan Representasi

- Pengetahuan adalah kumpulan fakta dari beberapa domain.
- Juga, Kami membutuhkan representasi "fakta" yang dapat dimanipulasi oleh suatu program.
- Selain itu, Bahasa Inggris Normal tidak cukup, terlalu sulit saat ini bagi program komputer untuk menarik kesimpulan dalam bahasa alami.
- Jadi beberapa representasi simbolis diperlukan.

Representasi pengetahuan yang baik memungkinkan akses yang cepat dan akurat ke pengetahuan dan pemahaman konten.

Sebuah sistem representasi pengetahuan harus memiliki sifat-sifat berikut.

1. Kecukupan Representasi

- Kemampuan untuk merepresentasikan semua jenis pengetahuan yang dibutuhkan dalam domain tersebut.

2. Kecukupan Inferensial
   - Juga, Kemampuan untuk memanipulasi struktur representasional untuk memperoleh struktur baru yang sesuai dengan pengetahuan baru yang disimpulkan dari yang lama.
3. Efisiensi Inferensial
   - Kemampuan untuk memasukkan informasi tambahan ke dalam struktur pengetahuan yang dapat digunakan untuk memfokuskan perhatian mekanisme inferensi ke arah yang paling menjanjikan.
4. Efisiensi Akuisisi
   - Selain itu, Kemampuan untuk memperoleh pengetahuan baru menggunakan metode otomatis sedapat mungkin daripada mengandalkan intervensi manusia.

Skema Representasi Pengetahuan

Pengetahuan Relasional

- Cara paling sederhana untuk merepresentasikan fakta deklaratif adalah sekumpulan relasi dari jenis yang sama yang digunakan dalam sistem database.
- Menyediakan kerangka kerja untuk membandingkan dua objek berdasarkan atribut yang setara. o Setiap contoh di mana dua objek yang berbeda dibandingkan adalah jenis pengetahuan relasional.
- Tabel di bawah ini menunjukkan cara sederhana untuk menyimpan fakta.
  - Juga, Fakta tentang sekumpulan objek diletakkan secara sistematis dalam kolom.
  - Representasi ini memberikan sedikit kesempatan untuk inferensi.

Tabel 3.1 cara sederhana untuk menyimpan fakta tentang sekumpulan objek diletakkan secara sistematis

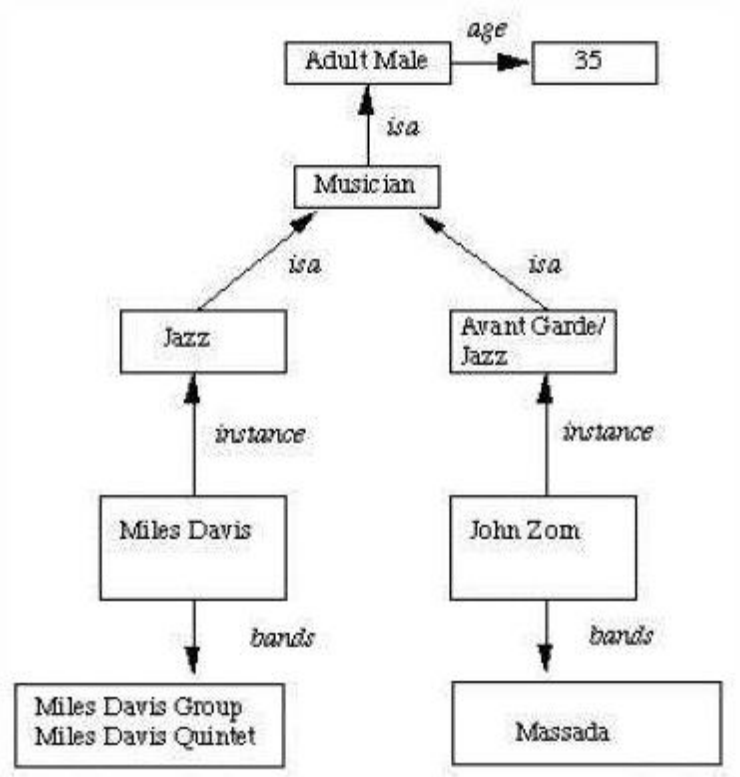
Pemain	Tinggi	Berat	Lemparan
Agus	6-0	180	Kanan-kanan
Budi	5-10	170	Kanan-kanan
Candra	6-2	215	Kiri-kiri
Dodik	6-3	205	Kiri-kana

Mengingat faktanya, tidak mungkin menjawab pertanyaan sederhana seperti: “Siapa pemain terberat?”

- Juga, Tetapi jika prosedur untuk menemukan pemain terberat disediakan, maka fakta-fakta ini akan memungkinkan prosedur itu untuk menghitung jawaban.
- Selain itu, Kita dapat menanyakan hal-hal seperti siapa “kelelawar – kiri” dan “melempar – ke kanan”.

Pengetahuan yang Dapat Diwariskan

- Di sini elemen pengetahuan mewarisi atribut dari orang tuanya.
- Pengetahuan yang terkandung dalam hierarki desain yang ditemukan dalam domain fungsional, fisik, dan proses.
- Dalam hierarki, elemen mewarisi atribut dari orang tuanya, tetapi dalam banyak kasus, tidak semua atribut elemen induk ditentukan ke elemen anak.
- Juga, Warisan adalah bentuk inferensi yang kuat, tetapi tidak memadai.
- Selain itu, KR dasar (Knowledge Representation) perlu ditambah dengan mekanisme inferensi.
- Warisan properti: Objek atau elemen kelas tertentu mewarisi atribut dan nilai dari kelas yang lebih umum.
- Jadi, Kelas-kelas diatur dalam hierarki umum.



Gambar 4.3 Pengetahuan yang Dapat Diwariskan

Node kotak — objek dan nilai atribut objek.

- Panah — titik dari objek ke nilainya.
- Struktur ini dikenal sebagai struktur slot dan pengisi, jaringan semantik atau kumpulan frame.

Langkah-langkah untuk mengambil nilai untuk atribut objek instance:

1. Temukan objek di basis pengetahuan
2. Jika ada nilai untuk atribut laporkan
3. Jika tidak, cari nilai instance, jika tidak ada yang gagal
4. Juga, Pergi ke node itu dan temukan nilai untuk atribut dan kemudian laporkan
5. Jika tidak, cari dengan menggunakan is sampai nilai ditemukan untuk atribut tersebut.

Pengetahuan Inferensial

- Pengetahuan ini menghasilkan informasi baru dari informasi yang diberikan.
- Informasi baru ini tidak memerlukan pengumpulan data lebih lanjut dari sumber formulir tetapi membutuhkan analisis informasi yang diberikan untuk menghasilkan pengetahuan baru.
- Contoh: diberikan seperangkat hubungan dan nilai, seseorang dapat menyimpulkan nilai atau hubungan lain. Logika predikat (pengurangan matematis) yang digunakan untuk menyimpulkan dari sekumpulan atribut.
- Selain itu, Inferensi melalui logika predikat menggunakan serangkaian operasi logis untuk menghubungkan data individual.
- Mewakili pengetahuan sebagai logika formal: Semua anjing memiliki ekor x: $dog(x) \rightarrow hastail(x)$

- Keuntungan:
  - Seperangkat aturan yang ketat.
  - Dapat digunakan untuk memperoleh lebih banyak fakta.
  - Juga, Kebenaran dari pernyataan baru dapat diverifikasi.
  - Dijamin kebenarannya.
- Jadi, Banyak prosedur inferensi yang tersedia untuk menerapkan aturan logika standar yang populer di sistem AI. misalnya pembuktian teorema otomatis.

Pengetahuan prosedural

- Sebuah representasi di mana informasi kontrol, untuk menggunakan pengetahuan, tertanam dalam pengetahuan itu sendiri. Misalnya, program komputer, petunjuk arah, dan resep; ini menunjukkan
- penggunaan atau implementasi tertentu;
- Selain itu, Pengetahuan dikodekan dalam beberapa prosedur, program kecil yang tahu bagaimana melakukan hal-hal tertentu, bagaimana memproses.

Keuntungan:

- Pengetahuan heuristik atau domain-spesifik dapat mewakili.
- Selain itu, inferensi logis yang diperluas, seperti penalaran default difasilitasi.
- Juga, efek samping dari tindakan dapat menjadi model. Beberapa aturan mungkin menjadi salah pada waktunya. Melacak ini dalam sistem besar mungkin rumit.

Kekurangan:

- Kelengkapan — tidak semua kasus dapat mewakili.
- Konsistensi — tidak semua pemotongan mungkin benar. misalnya Jika kita tahu bahwa Fred adalah seekor burung, kita dapat menyimpulkan bahwa Fred dapat terbang. Nanti kita mungkin menemukan bahwa Fred adalah anemu.
- Modularitas dikorbankan. Perubahan dalam basis pengetahuan mungkin memiliki efek yang luas.
- Informasi kontrol yang rumit.

Menggunakan Logika Predikat

Representasi Fakta Sederhana dalam Logika

Logika proposisi berguna karena sederhana untuk ditangani dan prosedur keputusan untuk itu ada. Juga, Untuk menarik kesimpulan, fakta direpresentasikan dengan cara yang lebih nyaman sebagai,

1. Ali adalah seorang pria.
   - laki-laki (Ali)
2. Ahmad adalah seorang pria.
   - manusia (Ahmad)
3. Semua manusia fana.
   - fana (laki-laki)

Tetapi logika proposisional gagal menangkap hubungan antara individu yang menjadi manusia dan individu yang fana.

- Bagaimana kalimat-kalimat ini dapat direpresentasikan sehingga kita dapat menyimpulkan kalimat ketiga dari dua kalimat pertama?
- Juga, logika Proposisional berkomitmen hanya pada keberadaan fakta yang mungkin atau mungkin tidak terjadi di dunia yang diwakili.
- Selain itu, ia memiliki sintaks yang sederhana dan semantik yang sederhana. Cukuplah untuk menggambarkan proses inferensi.
- Logika proposisi dengan cepat menjadi tidak praktis, bahkan untuk dunia yang sangat kecil.

logika predikat

Logika predikat orde pertama (FOPL) memodelkan dunia dalam hal:

- Obyek, yaitu hal-hal yang memiliki identitas individu
- Sifat-sifat benda yang membedakannya dari benda lain
- Hubungan yang ada di antara kumpulan objek
- Fungsi, yang merupakan bagian dari hubungan di mana hanya ada satu "nilai" untuk "masukan" yang diberikan

Logika predikat orde pertama (FOPL) menyediakan

- Konstanta: a, b, dog33. Beri nama objek tertentu.
- Variabel: X, Y. Merujuk pada suatu objek tanpa menamainya.
- Fungsi: Pemetaan dari objek ke objek.
- Istilah: Mengacu pada objek

- Kalimat Atom: in(ayah-of(X), food6) Bisa benar atau salah, Sesuai dengan proposisional simbol P, Q.

Formula yang terbentuk dengan baik (wff) adalah kalimat yang tidak mengandung variabel "bebas". Jadi, Artinya, semua variabel "terikat" oleh quantifier universal atau eksistensial. (∀x)P(x, y) memiliki x terikat sebagai variabel terkuantifikasi universal, tetapi y bebas.

Kuantifier

Kuantifikasi universal

- (∀x)P(x) berarti bahwa P berlaku untuk semua nilai x dalam domain yang terkait dengan variabel tersebut
- Mis., (∀x) lumba-lumba(x) → mamalia(x)
- Kuantifikasi eksistensial
- (∃ x)P(x) berarti bahwa P berlaku untuk beberapa nilai x dalam domain yang terkait dengan variabel itu
- Misalnya, (∃ x) mamalia(x) bertelur(x)

Juga, Perhatikan contoh berikut yang menunjukkan penggunaan logika predikat sebagai cara untuk merepresentasikan pengetahuan.

1. Marcus adalah seorang pria.
2. Marcus adalah seorang Pompeian.
3. Semua Pompeian adalah orang Romawi.
4. Caesar adalah seorang penguasa.
5. Juga, Semua Pompeian setia kepada Caesar atau membencinya.
6. Setiap orang setia pada seseorang.
7. Orang-orang hanya mencoba membunuh penguasa yang tidak mereka setiai.
8. Marcus mencoba membunuh Caesar.

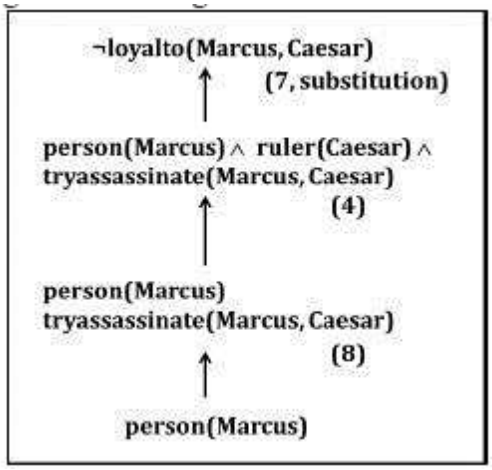
Fakta-fakta yang dijelaskan oleh kalimat-kalimat ini dapat direpresentasikan sebagai serangkaian formula yang terbentuk dengan baik (wffs) sebagai berikut:

1. Marcus adalah seorang pria.
   - laki-laki (Marcus)
2. Marcus adalah seorang Pompeian.
   - Pompeian (Marcus)
3. Semua Pompeian adalah orang Romawi.
   - x: Pompeian(x) → Roman(x)
4. Caesar adalah seorang penguasa.
   - penguasa (Kaisar)
5. Semua Pompeian setia kepada Caesar atau membencinya.
   - inklusif-atau
   - x: Roman(x) → loyalto(x, Caesar) benci(x, Caesar)
   - eksklusif-atau
   - x: Roman(x) → (loyalto(x, Caesar) benci(x, Caesar))
   - (¬loyalto(x, Caesar) benci(x, Caesar))
6. Setiap orang setia pada seseorang.
   - x: y: loyalto(x, y)
7. Orang-orang hanya mencoba membunuh penguasa yang tidak mereka setiai.
   - x: y: orang(x) penggaris(y) percobaan pembunuhan(x, y)
   - →¬loyalto(x, y)
8. Marcus mencoba membunuh Caesar.
   - mencoba membunuh (Marcus, Caesar)

Sekarang misalkan jika kita ingin menggunakan pernyataan ini untuk menjawab pertanyaan: Apakah Marcus setia kepada Caesar?

Juga, Sekarang mari kita coba menghasilkan bukti formal, dengan alasan mundur dari tujuan yang diinginkan: Ioyalto(Marcus, Caesar)

Untuk membuktikan tujuan, kita perlu menggunakan aturan inferensi untuk mengubahnya menjadi tujuan lain (atau mungkin serangkaian tujuan) yang dapat, pada gilirannya, ditransformasikan, dan seterusnya, sampai tidak ada tujuan yang belum terpenuhi yang tersisa.

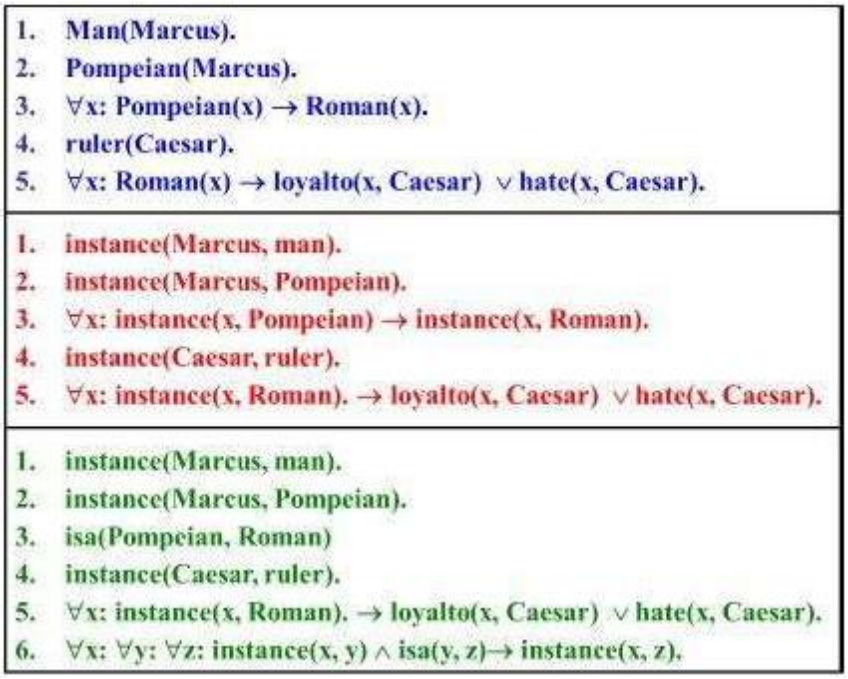


Gambar 4.4 Upaya untuk membuktikan loyalto(Marcus, Caesar).

- Masalahnya adalah, meskipun kita tahu bahwa Marcus adalah seorang pria, kita tidak memiliki cara untuk menyimpulkan bahwa Marcus adalah seorang manusia. Juga, Kita perlu menambahkan representasi fakta lain ke sistem kita, yaitu: man(x) → person(x)
- Sekarang kita dapat memenuhi tujuan terakhir dan menghasilkan bukti bahwa Marcus tidak setia kepada Caesar.
- Selain itu, Dari contoh sederhana ini, kita melihat bahwa tiga masalah penting harus ditangani dalam proses mengubah kalimat bahasa Inggris menjadi pernyataan logis dan kemudian menggunakan pernyataan tersebut untuk menyimpulkan yang baru:
  1. Banyak kalimat bahasa Inggris yang ambigu (misalnya 5, 6, dan 7 di atas). Memilih interpretasi yang benar mungkin sulit.
  2. Juga, sering ada pilihan bagaimana merepresentasikan pengetahuan. Representasi sederhana diinginkan, tetapi mereka mungkin mengecualikan jenis penalaran tertentu.
  3. Mirip, Bahkan dalam situasi yang sangat sederhana, satu set kalimat tidak mungkin mengandung semua informasi yang diperlukan untuk alasan tentang topik yang dihadapi. Agar dapat menggunakan seperangkat pernyataan secara efektif. Selain itu, biasanya diperlukan untuk memiliki akses ke kumpulan pernyataan lain yang mewakili fakta yang orang anggap terlalu jelas untuk disebutkan.

Mewakili Instance dan Hubungan ISA

- Instance dan isa atribut khusus memainkan peran penting khususnya dalam bentuk penalaran yang berguna yang disebut pewarisan properti.
- Instance predikat dan isa secara eksplisit menangkap hubungan yang mereka gunakan untuk mengekspresikan, yaitu keanggotaan kelas dan inklusi kelas.
- 4.2 menunjukkan lima kalimat pertama dari bagian terakhir yang direpresentasikan dalam logika dalam tiga cara berbeda.
- Bagian pertama dari gambar berisi representasi yang telah kita diskusikan. Dalam representasi ini, keanggotaan kelas diwakili dengan predikat unary (seperti Roman), yang masing-masing sesuai dengan kelas.
- Menyatakan bahwa P(x) benar sama dengan menyatakan bahwa x adalah turunan (atau elemen) dari P.
- Bagian kedua dari gambar berisi representasi yang menggunakan predikat instance secara eksplisit.



Gambar 4.5 Tiga cara merepresentasikan keanggotaan kelas: Hubungan ISA

- Instance predikat adalah biner, yang argumen pertamanya adalah objek dan argumen keduanya adalah kelas tempat objek tersebut berada.
- Tetapi representasi ini tidak menggunakan predikat isa eksplisit.
- Sebaliknya, hubungan subkelas, seperti antara Pompeian dan Romawi, dijelaskan seperti yang ditunjukkan dalam kalimat 3.
- Aturan implikasi menyatakan bahwa jika suatu objek adalah turunan dari subkelas Pompeian maka itu adalah turunan dari superkelas Roman.
- Perhatikan bahwa aturan ini setara dengan definisi teori himpunan standar dari hubungan subclasssuperclass.
- Bagian ketiga berisi representasi yang menggunakan predikat instance dan isa secara eksplisit.
- Penggunaan predikat isa menyederhanakan representasi kalimat 3, tetapi memerlukan satu aksioma tambahan (ditunjukkan di sini sebagai nomor 6).

Fungsi dan Predikat yang Dapat Dihitung

- Untuk menyatakan fakta sederhana, seperti hubungan lebih besar dari dan lebih kecil dari berikut: gt(1,O) It(0,1) gt(2,1) It(1,2) gt(3,2) It (2,3)
- Sering juga berguna untuk memiliki fungsi yang dapat dihitung serta predikat yang dapat dihitung. Jadi kita mungkin ingin dapat mengevaluasi kebenaran dari gt(2 + 3,1)
- Untuk melakukannya, pertama-tama kita harus menghitung nilai fungsi plus yang diberikan argumen 2 dan 3, lalu mengirim argumen 5 dan 1 ke gt.

Pertimbangkan serangkaian fakta berikut, sekali lagi melibatkan Marcus:

1) Ali adalah seorang pria.

pria (Ali)

2) Ali adalah seorang Pompeian.

Pompeian (Ali)

3) Ali lahir pada tahun 40 M.

lahir (Ali, 40)

4) Semua manusia adalah fana.

x: manusia(x) → manusia(x)

5) Semua orang Pompeian meninggal ketika gunung berapi meletus pada tahun 79 M.

meletus(gunung berapi, 79) x : [Pompeian(x) → mati(x, 79)]

6) Tidak ada manusia hidup lebih lama dari 150 tahun.

x: t1: At2: fana(x) lahir(x, t1) gt(t2 – t1,150) → mati(x, t2)

7) Sekarang tahun 1991.

sekarang = 1991

Jadi, contoh di atas menunjukkan bagaimana gagasan tentang fungsi dan predikat yang dapat dihitung ini dapat bermanfaat. Itu juga menggunakan gagasan kesetaraan dan memungkinkan objek yang sama untuk diganti masing-masing lainnya setiap kali tampaknya membantu untuk melakukannya selama pembuktian.

- Jadi, Sekarang misalkan kita ingin menjawab pertanyaan "Apakah Marcus masih hidup?"
- Pernyataan yang disarankan di sini, mungkin ada dua cara untuk menyimpulkan jawaban.
- Entah kita dapat menunjukkan bahwa Marcus mati karena dia terbunuh oleh gunung berapi atau kita dapat menunjukkan bahwa dia pasti mati karena dia akan berusia lebih dari 150 tahun, yang kita tahu tidak mungkin.
- Juga, segera setelah kami mencoba untuk mengikuti salah satu dari jalur tersebut dengan ketat, kami menemukan, seperti yang kami lakukan pada contoh terakhir, bahwa kami memerlukan beberapa pengetahuan tambahan. Misalnya, pernyataan kami berbicara tentang kematian, tetapi tidak mengatakan apa pun yang berhubungan dengan hidup, itulah pertanyaan yang diajukan. Jadi kami menambahkan fakta berikut:

8) Hidup berarti tidak mati.

x: t: [hidup(x, t) → mati(x, t)] [¬ mati(x, t) → hidup(x, t)]

9) Jika seseorang meninggal, maka dia mati di kemudian hari.

x: t1: At2: mati(x, t1) gt(t2, t1) → mati(x, t2)

Jadi, Sekarang mari kita coba menjawab pertanyaan "Apakah Marcus masih hidup?" dengan membuktikan: hidup (Marcus, sekarang)

Resolusi

Resolusi Proposisional

1. Ubah semua proposisi F menjadi bentuk klausa.
2. Ubahlah P kedalam menjadi bentuk klausa dan Negasikan. Tambahkan ke set klausa yang diperoleh pada langkah 1.
3. Ulangi sampai kontradiksi ditemukan atau tidak ada kemajuan yang dapat dibuat:
    1. Pilih dua klausa. Sebut ini klausa induk.
    2. Bersama-sama untuk menyelesaikan. Klausa yang dihasilkan, disebut resolvent, akan menjadi disjungsi dari semua literal kedua klausa induk dengan pengecualian berikut: Jika ada pasangan literal L dan L sedemikian rupa sehingga salah satu klausa induk berisi L dan lainnya mengandung L, kemudian pilih salah satu pasangan tersebut dan hilangkan L dan L dari pelarut.
    3. Jika penyelesainya adalah klausa kosong, maka telah ditemukan kontradiksi. Jika tidak, maka tambahkan ke set kelas yang tersedia untuk prosedur.

Algoritma Unifikasi

- Dalam logika proposisional, mudah untuk menentukan bahwa dua literal tidak dapat keduanya benar pada saat yang sama.
- Cukup mencari L dan L dalam logika predikat, proses pencocokan ini lebih rumit karena argumen predikat harus dipertimbangkan.
- Misalnya, man(John) dan man(John) adalah kontradiksi, sedangkan man(John) dan man(Spot) tidak.
- Jadi, untuk menentukan kontradiksi, kita memerlukan prosedur pencocokan yang membandingkan dua literal dan menemukan apakah ada himpunan substitusi yang membuatnya identik.
- Ada prosedur rekursif langsung, yang disebut algoritma unifikasi, yang melakukannya.

Algoritma: Unify(L1, L2)

1. Jika L1 atau L2 keduanya variabel atau konstanta, maka:
    1. Jika L1 dan L2 identik, maka kembalikan NIL.

2. Lain jika L1 adalah variabel, maka jika L1 terjadi di L2 maka kembali {FAIL}, yang lain kembali (L2/L1).
3. Juga, Else jika L2 adalah variabel, maka jika L2 terjadi di L1 maka kembalikan {FAIL}, lain kembalikan (L1/L2). d. Jika tidak, kembalikan {GAGAL}.

2. Jika simbol predikat awal pada L1 dan L2 tidak identik, maka kembalikan {FAIL}.
3. Jika LI dan L2 memiliki jumlah argumen yang berbeda, maka kembalikan {FAIL}.
4. Atur SUBST ke NIL. (Pada akhir prosedur ini, SUBST akan berisi semua substitusi yang digunakan untuk menyatukan L1 dan L2.)
5. Untuk I
    1. dengan banyaknya argumen di L1 :
        1. Panggil Unify dengan argumen ke-i dari L1 dan argumen ke-i dari L2, letakkan hasilnya di S.
        2. Jika S berisi FAIL maka kembalikan {FAIL}.
        3. Jika S tidak sama dengan NIL maka:
    2. Terapkan S ke sisa L1 dan L2.
    3. SUBST: = TAMBAH(S, SUBST).
6. Kembalikan SUBST.

Resolusi dalam Logika Predikat

Kami sekarang dapat menyatakan resolusi untuk solusi logis sebagai berikut, dengan asumsi satu set pernyataan yang diberikan F dan pernyataan harus dibuktikan P: Algoritma: Resolusi

1. Ubah semua pernyataan F menjadi bentuk klausa.
2. Abaikan P dan ubah hasilnya menjadi bentuk klausa. tambahkan ke set klausa yang diperoleh di 1.
3. sampai kontra ditemukan, tidak ada kemajuan yang dapat dibuat, atau jumlah usaha yang telah ditentukan telah diperpanjang.
    1. Pilih dua klausa. Sebut ini klausa induk.
    2. Kerjakan bersama. Putuskan untuk memisahkan semua literal dari dua klausa induk dengan substitusi yang sesuai dilakukan dan dengan perkembangan berikut: Jika ada satu pasangan literal T1 dan T2, salah satu klausa induk berisi T2 dan yang lainnya berisi T1 dan jika T1 dan T2 tidak dapat dipisahkan, maka
    3. baik T1 maupun T2 tidak akan muncul dalam pelarut. Kami memanggil T1 dan T2
    4. Melengkapi literal. Gunakan substitusi yang dihasilkan oleh unifikasi untuk membuat pelarut. Jika ada lebih dari satu pasangan literal komplementer, hanya satu pasangan yang harus dikeluarkan dari pelarut.
4. Jika solusinya adalah klausa kosong, maka telah ditemukan kontradiksi. Selain itu, Jika tidak, tambahkan ke set kelas yang tersedia untuk prosedur.

Prosedur Resolusi

- Resolusi adalah prosedur, yang memperoleh efisiensi dari fakta bahwa ia beroperasi pada pernyataan yang telah diubah ke bentuk standar yang sangat nyaman.
- Resolusi menghasilkan bukti dengan sanggahan.
- Dengan kata lain, untuk membuktikan suatu pernyataan (yaitu, untuk menunjukkan bahwa itu valid), resolusi mencoba untuk menunjukkan bahwa negasi dari pernyataan tersebut menghasilkan kontradiksi dengan pernyataan yang diketahui (yaitu, tidak memuaskan).

Prosedur resolusi adalah proses iteratif sederhana: pada setiap langkah, dua klausa, yang disebut klausa induk, dibandingkan (diselesaikan), menghasilkan klausa baru yang telah disimpulkan darinya. Klausa baru mewakili cara dua klausa induk berinteraksi satu sama lain. Misalkan ada dua klausa dalam sistem:
musim dingin V musim panas
musim dingin V dingin

- Sekarang kita amati bahwa tepat salah satu musim dingin dan musim dingin akan benar pada setiap titik.
- Jika musim dingin benar, maka dingin harus benar untuk menjamin kebenaran klausa kedua. Jika musim dingin benar, maka musim panas harus benar untuk menjamin kebenaran klausa pertama.

- Jadi kita melihat bahwa dari dua klausa ini kita dapat menyimpulkan musim panas V dingin
- Ini adalah deduksi yang akan dibuat oleh prosedur penyelesaian.
- Resolusi beroperasi dengan mengambil dua klausa yang masing-masing berisi literal yang sama, dalam hal ini

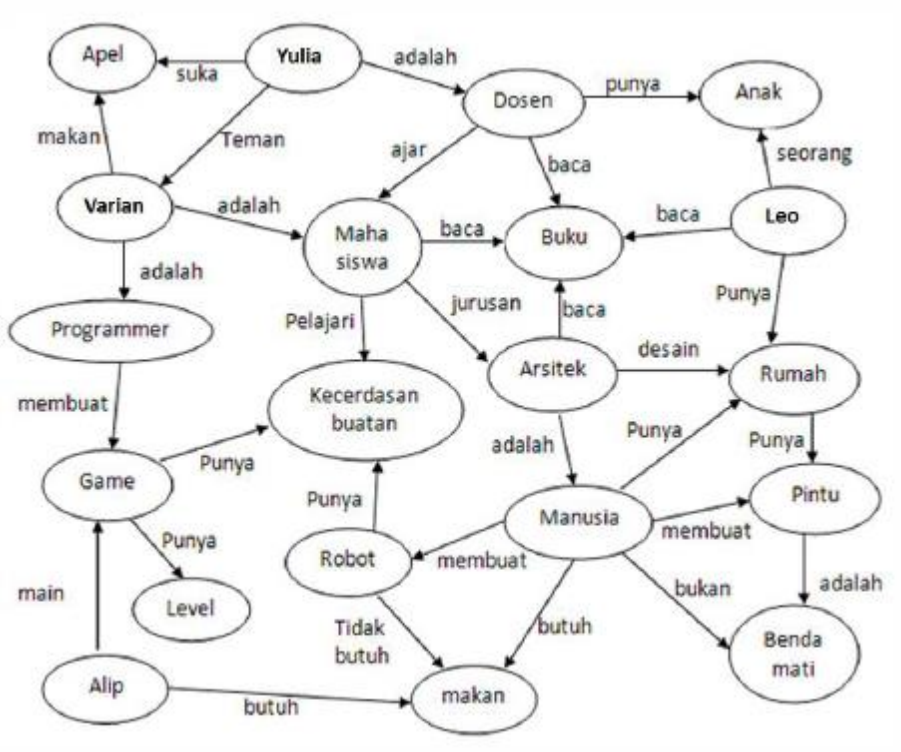
contoh, musim dingin.

- Selain itu, literal harus muncul dalam bentuk positif di satu klausa dan dalam bentuk negatif di klausa lainnya. Resolven diperoleh dengan menggabungkan semua literal dari dua klausa induk kecuali yang membatalkan.
- Jika klausa yang dihasilkan adalah klausa kosong, maka telah ditemukan kontradiksi.

Misalnya, dua klausa
musim dingin
musim dingin
akan menghasilkan klausa kosong.

Jaringan Semantik

Pengetahuan diatur dalam jaringan yang memiliki komponen utama sebagai berikut:
- Simpul atau Node: mewakili objek, konsep, atau situasi. Dinyatakan dengan persegi atau lingkaran
- Arcs/Link: Mendeklarasikan hubungan antar node. Ditunjukkan oleh panah.



Gambar 4.6 Contoh jaringan Semantik

Frame

Frames: merupakan semantic net dilengkapi dengan properties. Suatu Frame menggambarkan entitas sebagai set dari attribute dan nilai yang bersesuaian.Suatu frame dapat berelasi dengan frame yang lainnya.

Tiga komponen utama dari frame
• frame name
• attributes (slots)
• values (subslots)

Book Frame
Slot à *Subslots*
Title à *AI. A modern Approach* Author à *Russell & Norvig* Year à *2003*

Contoh di atas dibentuk dalam Frame:

Nama Frame: Burung	
Orangtua	Hewan
Anak	Kenari, Pipit
Mempunyai	Sayap
Cara Berpindah tempat	Terbang

Contoh:
Bingkai Pohon
Spesialisasi dalam: Tanaman
Jumlah bar: integer (default 1)
Jenis kulit: halus
Model daun: jenis konifera, gugur
Bentuk daun: sederhana, melengkung, campuran
Bingkai Pohon Semak
Keistimewaan: Pohon
Jumlah batang: 3
Jenis kulit: halus
Gaya daun: mengubah daun
Bentuk daun: sederhana, melengkung

Naskah (Script)
Ketergantungan konseptual adalah teori tentang bagaimana merepresentasikan pengetahuan tentang peristiwa yang biasanya terkandung dalam kalimat bahasa alami.
Contoh: Representasi Ketergantungan Konseptual
"Budi memberi Atika sebuah buku"
Naskah adalah representasi pengetahuan yang menggambarkan urutan peristiwa. Naskah dilengkapi dengan elemen untuk memudahkan memahami urutan kejadian.
sebuah. Trek: kemungkinan variasi dalam skrip
b. Kondisi Input: situasi yang harus dipenuhi sebelum sesuatu dapat terjadi
c. Props/Supporters: objek pendukung yang digunakan dalam urutan kejadian yang terjadi
d. Peran: orang-orang yang terlibat dalam suatu peran
e. Adegan: urutan kejadian yang sebenarnya
f. Hasil: keadaan akhir yang terjadi setelah urutan kejadian dalam naskah terjadi
Contoh script acara saat "Memesan Makanan"
Script : Food Delivery Track (track): pengiriman makanan melalui telepon Peran (role): pelanggan, call service, chef, kasir, kurir Prop (support): restoran, telepon, komputer (database), peralatan masak, bahan bahan masakan, kendaraan Kondisi input: pelanggan memesan nomor dengan memberi tahu nama, alamat, dan telepon ke call center
Adegan (adegan) – 1: Pelanggan memesan makanan
• Pelanggan menelepon restoran tujuan
• Pelanggan menyebutkan menu yang ingin dipesan

• Call service untuk mengecek ketersediaan menu yang dipesan
• Pelanggan menyebutkan nama, alamat dan nomor telepon call center
• Call center menyebutkan jumlah menu yang dipesan dan biayanya

Adegan – 2: Restoran menyiapkan pesanan
• Koki memasak menu makanan yang dipesan
• Koki dengan rapi membungkus paket menu yang dipesan
• Kasir membuat kwitansi pesanan dan menyerahkannya ke kurir
• Koki memberikan menu paket ke kurir

Adegan – 3: Kurir mengantarkan paket pesanan
• Kurir menerima dari menu paket chef yang dipesan
• Kurir membawa kwitansi yang berisi nama, alamat, dan nomor telepon pelanggan, beserta jumlah yang akan ditentukan oleh pelanggan
• Kurir memasukkan paket pesanan ke dalam kotak
• Departemen kurir mencari alamat pemesan
• Kurir tiba di tempat tujuan
• Kurir menanyakan alamat pelanggan yang benar
Adegan – 4: Pelanggan menerima pesanan
• Pelanggan menerima paket pesanan dan tanda terima dari kurir
• Pelanggan melakukan pengecekan paket, apakah sesuai dengan pesanan atau tidak
• Pelanggan membayar paket pesanan ke kurir
• Pelanggan memberikan tip kepada kurir
• Kurir menerima pembayaran

Hasil :
• Kurir senang dan senang
• Selamat memesan
• Pesan penuh
• Pelanggan yang puas
• Pelanggan yang kecewa
• Pelanggan masih lapar

Aturan Produksi (Aturan Produksi)
Pengetahuan dalam aturan produksi direpresentasikan dalam bentuk
JIKA [kondisi] MAKA [Tindakan]
JIKA [premis] MAKA [Kesimpulan]
Production Rules adalah salah satu representasi pengetahuan yang menghubungkan premis dengan kesimpulan.
Bentuk: Jika premis maka kesimpulannya
Kesimpulan pada bagian adalah layak jika premis pada bagian jika layak adalah benar.
Contoh:
Jika hari ini hujan maka saya tidak kuliah.

4.2 Pengetahuan Prosedural vs Deklaratif

Pengetahuan Prosedural
Representasi di mana informasi kontrol yang diperlukan untuk menggunakan pengetahuan tertanam dalam pengetahuan itu sendiri misalnya program komputer, petunjuk arah, dan resep; ini menunjukkan penggunaan atau implementasi khusus. Perbedaan nyata antara pandangan pengetahuan deklaratif dan prosedural terletak pada di mana informasi kontrol berada. Sebagai contoh, perhatikan berikut ini Pria (Marcus) Pria (Kaisar) Orang (Cleopatra)
x: Pria(x) → Orang(x)
Sekarang coba jawab pertanyaannya.

?Orang(y) Basis pengetahuan membenarkan salah satu dari jawaban berikut.
Y=Marcus Y=Kaisar Y = Cleopatra

- Kami mendapatkan lebih dari satu nilai yang memenuhi predikat.
- Jika hanya satu nilai yang dibutuhkan, maka jawaban atas pertanyaan tersebut akan tergantung pada urutan asersi yang diperiksa selama pencarian jawaban.
- Jika pernyataan deklaratif maka mereka sendiri tidak mengatakan apa-apa tentang bagaimana mereka akan diperiksa. Dalam hal representasi prosedural, mereka mengatakan bagaimana mereka akan memeriksa. Pengetahuan Deklaratif
- Pernyataan di mana pengetahuan ditentukan, tetapi penggunaan pengetahuan itu tidak diberikan.
- Misalnya undang-undang, nama orang; inilah fakta-fakta yang dapat berdiri sendiri, tidak bergantung pada pengetahuan lain;
- Jadi untuk menggunakan representasi deklaratif, kita harus memiliki program yang menjelaskan apa yang harus dilakukan dengan pengetahuan dan bagaimana caranya.
- Sebagai contoh, sekumpulan asersi logis dapat digabungkan dengan pembuktian teorema resolusi untuk memberikan program lengkap untuk memecahkan masalah tetapi dalam beberapa kasus, asersi logis dapat dilihat sebagai program daripada data ke program.
- Oleh karena itu pernyataan implikasi menentukan jalur penalaran yang sah dan pernyataan otomatis memberikan titik awal dari jalur tersebut.
- Jalur ini menentukan jalur eksekusi yang mirip dengan 'jika kemudian lain' dalam pemrograman tradisional.
- Jadi pernyataan logis dapat dilihat sebagai representasi prosedural dari pengetahuan.

4.3 Logic Programming

- Pemrograman logika adalah paradigma pemrograman di mana pernyataan logis dipandang sebagai program.
- Ini adalah beberapa sistem pemrograman logika, PROLOG adalah salah satunya.
- Program PROLOG terdiri dari beberapa pernyataan logis di mana masing-masing adalah klausa tanduk
- yaitu klausa dengan paling banyak satu literal positif.
- Contoh : P, P V Q, P → Q
- Fakta-fakta terwakili pada Klausul Tanduk karena dua alasan.
  1. Karena representasi yang seragam, juru bahasa yang sederhana dan efisien dapat menulis.
  2. Logika Klausul Tanduk dapat ditentukan.
- Juga, Dua perbedaan pertama adalah fakta bahwa program PROLOG sebenarnya adalah kumpulan klausa Horn yang telah diubah sebagai berikut:-
  1. Jika Klausa Tanduk tidak mengandung literal negatif maka biarkan apa adanya.

  2. Juga, Jika tidak, tulis ulang klausa Horn sebagai implikasi, gabungkan semua literal negatif ke dalam anteseden implikasi dan literal positif tunggal ke dalam konsekuen.
- Selain itu, Prosedur ini menyebabkan klausa yang semula terdiri dari disjungsi literal (salah satunya positif) diubah menjadi implikasi tunggal yang antesedennya adalah konjungsi yang dikuantifikasi secara universal.
- Tetapi ketika kita menerapkan transformasi ini, setiap variabel yang terjadi dalam literal negatif dan sekarang terjadi pada anteseden menjadi terkuantifikasi secara eksistensial, sedangkan variabel dalam konsekuen masih terkuantifikasi secara universal.
- Misalnya klausa PROLOG P(x): – Q(x, y) sama dengan ekspresi logika x: y: Q (x, y) → P(x).
- Perbedaan antara logika dan representasi PROLOG adalah bahwa interpretasi PROLOG memiliki strategi kontrol yang tetap. Jadi, pernyataan dalam program PROLOG menentukan jalur pencarian tertentu untuk menjawab pertanyaan apa pun.
- Namun, asersi logis hanya mendefinisikan kumpulan jawaban tetapi bukan tentang bagaimana memilih di antara jawaban-jawaban itu jika ada lebih dari satu.

Perhatikan contoh berikut:

1. Representasi logis

x : pet(x) kecil (x) → apartmentpet(x)
x : kucing(x) anjing(x) → peliharaan(x)
x : pudel (x) → anjing (x) pudel kecil (x) (berbulu)

2. Representasi prolog

apartemenpet (x) : hewan peliharaan(x), hewan peliharaan kecil (x) (x): kucing (x)
hewan peliharaan (x): anjing (x)
anjing(x): pudel (x) kecil (x): pudel(x) pudel (berbulu)

4.4 Forward versus Backward Reasoning

Prosedur pencarian harus menemukan jalur antara keadaan awal dan tujuan. Ada dua arah di mana proses pencarian bisa dilanjutkan. Kedua jenis pencarian tersebut adalah:

1. Pencarian teruskan yang dimulai dari status awal
2. Pencarian mundur yang dimulai dari status tujuan

Sistem produksi memandang maju dan mundur sebagai proses simetris. Pertimbangkan permainan bermain 8 teka-teki. Aturan yang ditentukan adalah

Kotak 1 kosong dan kotak 2 berisi ubin n. →

- Juga, Kotak 2 kosong dan kotak 1 berisi ubin n. Kotak 1 kosong Kotak 4 berisi ubin n. →
- Juga, Kotak 4 kosong dan Kotak 1 berisi ubin n.

Kita dapat memecahkan masalah dengan 2 cara:

1. Alasan maju dari keadaan awal

- Langkah 1. Mulailah membangun pohon urutan gerakan dengan memulai konfigurasi awal pada akar pohon.
- Langkah 2. Hasilkan level pohon berikutnya dengan mencari semua aturan yang ruas kirinya cocok dengan simpul akar. Sisi kanan digunakan untuk membuat konfigurasi baru.
- Langkah 3. Hasilkan level berikutnya dengan mempertimbangkan node di level sebelumnya dan menerapkannya ke semua aturan yang sisi kirinya cocok.

2. Penalaran mundur dari tujuan menyatakan:

- Langkah 1. Mulailah membangun pohon urutan gerakan dengan memulai dengan konfigurasi simpul tujuan di akar pohon.
- Langkah 2. Hasilkan level pohon berikutnya dengan mencari semua aturan yang ruas kanannya cocok dengan simpul akar. Sisi kiri digunakan untuk membuat konfigurasi baru.
- Langkah 3. Hasilkan level berikutnya dengan mempertimbangkan node di level sebelumnya dan menerapkannya ke semua aturan yang sisi kanannya cocok.
- Jadi, Aturan yang sama dapat digunakan dalam kedua kasus.
- Juga, Dalam penalaran penerusan, sisi kiri aturan dicocokkan dengan status saat ini dan sisi kanan yang digunakan untuk menghasilkan status baru.
- Selain itu, Dalam penalaran mundur, sisi kanan aturan yang dicocokkan dengan keadaan saat ini dan ruas kiri digunakan untuk menghasilkan keadaan baru.

Ada empat faktor yang mempengaruhi jenis penalaran. Mereka,

1. Apakah ada lebih banyak kemungkinan keadaan awal atau tujuan? Kami pindah dari set yang lebih kecil ke set yang panjang.
2. Ke arah manakah faktor percabangan lebih besar? Kami melanjutkan ke arah dengan faktor percabangan yang lebih rendah.
3. Apakah program akan diminta untuk membenarkan proses penalarannya kepada pengguna? Jika, maka dipilih karena sangat dekat dengan cara berpikir pengguna.
4. Peristiwa seperti apa yang akan memicu episode pemecahan masalah? Jika itu adalah kedatangan faktor baru, penalaran ke depan masuk akal. Jika itu adalah kueri yang menginginkan respons, penalaran mundur lebih alami.

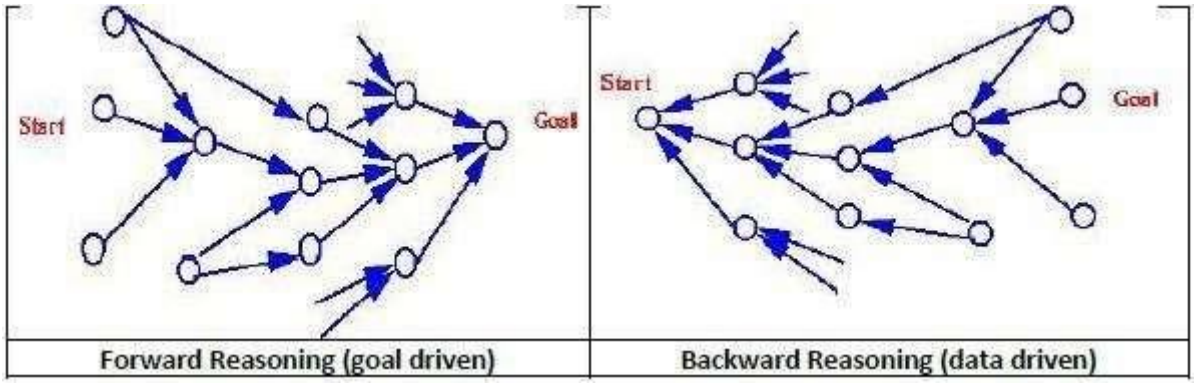
Contoh 1 Penalaran Maju versus Mundur

- Lebih mudah berkendara dari tempat asing dari rumah, daripada dari rumah ke tempat asing. Juga, Jika Anda menganggap rumah sebagai tempat awal sebagai tempat asing sebagai tujuan maka kita harus mundur dari tempat asing ke rumah.
- Contoh 2 Penalaran Maju versus Mundur

- Pertimbangkan masalah integrasi simbolik. Selain itu, Ruang masalah adalah seperangkat rumus, yang berisi ekspresi integral. Di sini MULAI sama dengan rumus yang diberikan dengan beberapa integral. GOAL setara dengan ekspresi rumus tanpa integral apa pun. Di sini kita mulai dari rumus dengan beberapa integral dan melanjutkan ke ekspresi bebas integral daripada mulai dari ekspresi bebas integral.
- Contoh 3 Penalaran Maju versus Mundur
- Faktor ketiga tidak lain adalah memutuskan apakah proses penalaran dapat membenarkan penalarannya. Jika itu membenarkan maka itu bisa diterapkan. Misalnya, dokter biasanya tidak mau menerima saran dari proses diagnosa karena tidak bisa menjelaskan alasannya.
- Contoh 4 Penalaran Maju versus Mundur
- Prolog adalah contoh dari sistem aturan rantai mundur. Dalam aturan Prolog terbatas pada klausa Horn. Hal ini memungkinkan untuk pengindeksan cepat karena semua aturan untuk menyimpulkan fakta yang diberikan berbagi kepala aturan yang sama. Aturan cocok dengan prosedur penyatuan. Unifikasi mencoba menemukan satu set ikatan untuk variabel untuk menyamakan sub-tujuan dengan kepala beberapa aturan. Aturan dalam program Prolog cocok dengan urutan kemunculannya.

Menggabungkan Penalaran Maju dan Mundur

- Alih-alih mencari maju atau mundur, Anda dapat mencari keduanya secara bersamaan.
- Juga, Yaitu, mulai maju dari keadaan awal dan mundur dari keadaan tujuan secara bersamaan hingga jalur bertemu.
- Strategi ini disebut pencarian dua arah. Gambar berikut menunjukkan alasan pencarian dua arah menjadi tidak efektif.



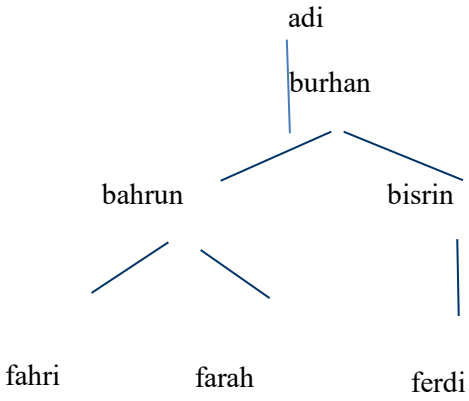
Gambar 4.7 Menggabungkan Penalaran Maju dan Mundur

Penalaran Maju versus Mundur

- Juga, Kedua pencarian dapat melewati satu sama lain sehingga menghasilkan lebih banyak pekerjaan.
- Berdasarkan bentuk aturan, seseorang dapat memutuskan apakah aturan yang sama dapat diterapkan baik untuk penalaran maju maupun mundur.
- Selain itu, Jika sisi kiri dan kanan aturan berisi pernyataan murni maka aturan dapat dibalik.
- Jadi aturan yang sama dapat berlaku untuk kedua jenis penalaran.
- Jika sisi kanan aturan berisi prosedur arbitrer maka aturan tidak dapat dibalik.
- Jadi, Dalam hal ini, saat menulis aturan, komitmen terhadap arah penalaran harus dibuat.

Soal :

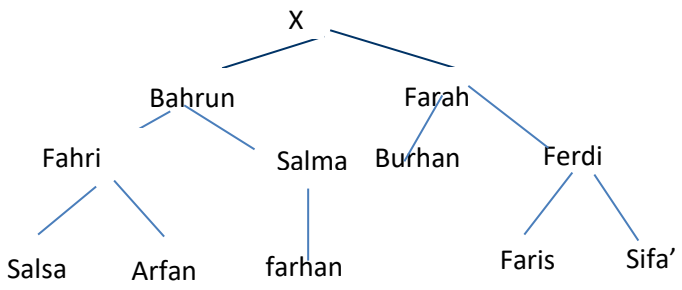
1. Misalkan pada sebuah Instasi terdapat tree sebagai berikut :



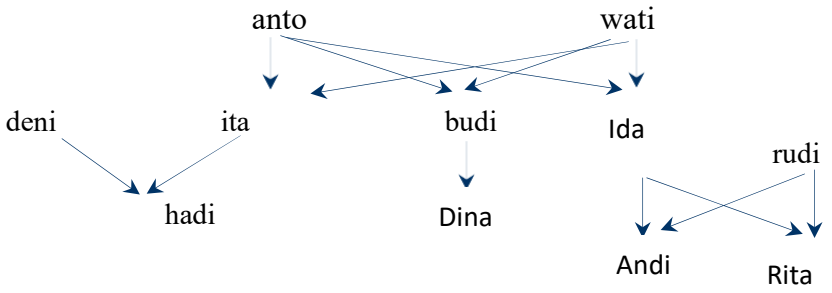
Pada pohon tersebut dapat dibaca bahwa Burhan adalah bawahan langsung Adi, sedangkan Adi adalah atasan langsung Burhan. Fahri dan Farah adalah bawahan Bahrun, sedangkan Fahri, Farah, Bahrun, Ferdi, Bisrin semuanya bawahan Burhan.

a. sebuah. Menggunakan sintaks di Prolog, buat representasi pengetahuan dari fakta di atas. (Dari definisi bawahan langsung).
b. b. Dengan menggunakan sintaks definisi bawahan langsung di atas, terjemahan untuk atasan langsung.
c. c. Bagaimana kita membuat pertanyaan tentang siapa bawahan langsung Burhan
d. d. Dengan menggunakan sintaks deskripsi secara rekursif untuk merepresentasikan fakta bawahan.

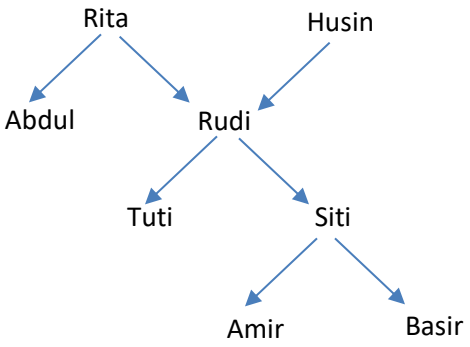
2. Seorang raja X dengan garis keturunan seperti di bawah ini hanya mencari satu dari keturunannya yang dapat menjadikan dirinya seorang raja. Tentu ada syarat untuk menjadi calon raja, yaitu ia adalah keturunan laki-laki atau keturunan laki-laki dari keturunan laki-laki. Dengan menggunakan representasi logis dan deskripsi rekursif raja X untuk mencari hanya keturunan pendukungnya untuk mengetahui siapa dia.



3. Dari silsilah keluarga di bawah ini, representasi logis yang menyatakan predikat putra, putri, pria, wanita. Kemudian dari predikat ini buatlah hubungan orang tua, saudara laki-laki, saudara perempuan, paman, bibi, kakek, nenek:



4. Dari silsilah keluarga di bawah ini, rumuskan dalam Prolog sebuah pertanyaan tentang:
   a. sebuah. Siapa orang tua Basir?
   b. Siapa nenek Siti?
   c. Apakah Tuti punya anak?

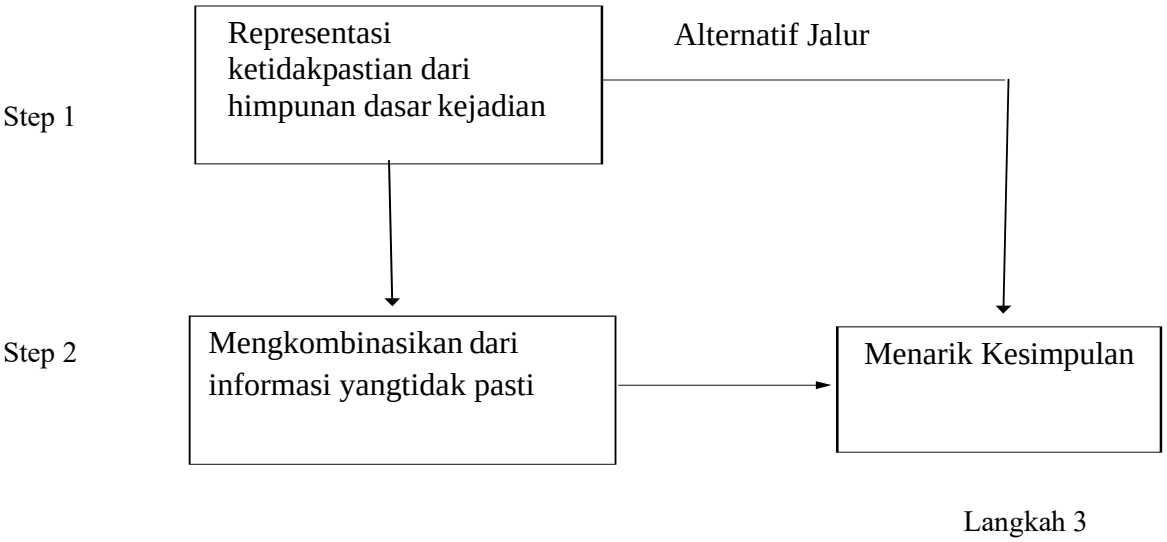


BAB 5
Teori Ketidakpastian

Uncertainty di dalam Kecerdasan buatan ditampilkan dalam tiga proses.

a. Seorang ahli atau pakar yang memberikan pengetahuan yang tidak pasti (salah), berupa aturan atau istilah yang mempunyai nilai dalam bentuk probabilitas. Nilai tersebut bisa berupa numerik (misalnya, nilai probabilitas), simbolis, atau grafik.
b. Pengetahuan yang tidak pasti tentang himpunan peristiwa dasar dalam menarik kesimpulan untuk kasus-kaus sederhana dapat langsung digunakan.
c. Untuk menarik kesimpulan dapat digunakan sistem berbasis pengetahuan.

Proses Uncertainty:



Gambar 5.1 Langkah – langkah Ketidakpastian

Jalur Alternatif

Terdapat studi kasus, berbagai peristiwa yang saling berhubungan. Sehingga diperlukan semacam penggabungan sebuah petunjuk atau informasi diberikan pada langkah 1 ke dalam sistem nilai tujuan (goal value).

Dapat dilakukan secara terintegrasi dari beberapa metode. Probabilitas Bayesian merupakan metode utama, teori kejadian, faktor kepastian, dan himpunan logika fuzzy.

Asal dari Uncertainty

1. Data, bisa berupa data yang tidak bisa diandalkan, kehilangan sebuah data, data yang tidak sesuai untuk disajikan, data yang tersimpan, ketidak konsisten sebuah data, data yang bersifat subjektif, data yang didapat dari kelalaian.
2. Pengetahuan dari para ahli atau Pakar
   a.. Inkonsistensi antara para ahli yang berbeda beda
   b. Kewajaran ("tebakan terbaik" dari ahli)
   c. Kwalitas

i. Pemahaman mendalam tentang sebab dan akibat atau pengetahuan kausal
ii. Kwalitas secara pengamatan atau statistik

d. Cakupan

asal dari Ketidakpastian juga dapat diperoleh dari :

1. representasi pengetahuan

a. Pada sistem nyata terjadinya keterbatasan model

b. Penggunaan mekanisme representasi dilakukan dengan terbatas

2. Proses untuk Inferensi

a. Deduktif merupakan hasil diperoleh dengan formal, namun mempunyai kesalahan di sistem yang ada

b. Induktif - Kesimpulan yang baru tidak diperoleh dengan benar

c. Metode penalaran tidak sehat atau tidak valid

Ketidakpastian Representasi

Ada 3 metode yang dasar dalam merepresentasikan sebuah ketidakpastian yaitu :

1. numerik,

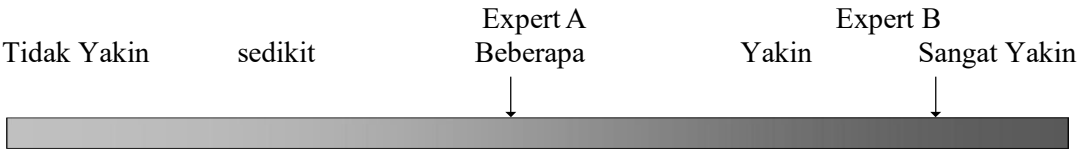
Representasi Numerik

a. Cara atau metode yang umum untuk menggambarkan ketidakpastian yaitu numerik, menggunakan skala 2 angka ekstrem. Misalnya, angka 0 digunakan sebagai wakil tidak pasti sementara angka 1 atau angka 100 mewakili angka yang pasti.
b. Penggunaan nomor ini mengakibatkan adanya kesulitan yang berakibat munculnya bias. Misalnya, para ahli atau pakar mengilustrasikan angka untuk pengamatan mereka berdasarkan persepsi mereka, yang mungkin berbeda dari ahli lainnya.

2. grafik,

Grafik Representasi

Umumnya grafik berbentuk batang mendatar, misalnya ekspresi pakar tentang tingkat kepercayaannya terhadap adanya sebuah kejadian.



1. simbolik.

Representasi dari Simbolik

a. ada beberapa cara yang digunkan untuk merepresentasikan ketidakpastian dalam istilah simbolis. Sebagian besar ahli menggunakan pendekatan skala Likert untuk mengungkapkan pendapat mereka.
b. Misalnya, ahli akan menanyakan apa yang Anda sukai tentang sesuatu pada skala 3 poin, yaitu; tidak suka, netral, dan suka.
c. Metode yang lain adalah fuzzy logik – akan segera dijelaskan
d. Representasi yang berupa simbolik umumnya menggabungkan dari metode – metode yang ada.

Masalah tentang Probabilitas

a. Probabilitas dapat dinyatakan dengan derajat interval dari kepercayaan dalam sebuah premis atau kesimpulan.

b. Probabilitas merupakan sebuah peluang suatu event atau kejadian tertentu yang akan terjadi atau yang akan tidak terjadi. Nilai probabilitas dihitung melalui formula sebagai berikut:

$$P(X) = \frac{Jumlah\ kejadian\ yang\ terjadi}{jumlah\ total\ kejadian} \quad (1)$$

c. P(X) atau Peluang suatu kejadian X adalah perbandingan banyaknya kejadian yang ada pada X dengan jumlah saat itu.
d. Di beberapa sistem nilai probabilitas akan terjadi. Misalnya, aturan yang memiliki 3 bagian dalam pendahuluan, Nilai probabilitas untuk masing-masing. Sebagai produk dari probabilitas individu maka probabilitas total aturan dapat dihitung. Bila bagian-bagian dari sebelumnya independen satu sama lain. Jika setiap probabilitas dengan nilai 0,8 ; 0,6; dan 0,55; maka peluang untuk totalnya P = (0.8)(0.6)(0.55) = 0.264
e. Probabilitas mempunyai nilai kurang lebih 41 persen. Namun perolehan ini akan baik dan layak bila nilai dari probabilitas individu pada bagian sebelumnya tidak saling mempengaruhi.
f. Beberapa pendekatan pada sistem dapat dilakukan untuk penggabungan probabilitas dapat dilakukan.
g. Misalnya, probabilitas gabungan atau dapat dikalikan atau dirata-ratakan (menggunakan rata-rata sederhana atau tertimbang). Bila setiap probabilitas saling ketergantungan atau bergantung dalam sistem, kita dapat menggunakan metode Bayes.

5.1 Teorema Bayes

a. Teorema Bayes merupakan mekanisme untuk menggabungkan peristiwa baru dan peristiwa yang dapat dinyatakan kedalam probabilitas subjektif.
b. Teorema bayes ini dapat dipergunakan untuk meninjau probabilitas awal atau probabilitas sebelumnya menurut informasi baru. Hasilnya disebut probabilitas posterior atau probabiltas akhir.
c. Dua probabilitas diketahui, untuk sebuah kasus yang cukup sederhana. Satu digunakan untuk kejadian yang terjadi pada A dan yang lainnya untuk kejadian yang terjadi pada B.

$$P(X/Y) = \frac{\mathrm{P(Y/X) * P(X)}}{\mathrm{P(Y/X)\ P(X) + P(Y/\neg X) * P(\neg X)}} \quad (2)$$

Di mana :
P(X/Y) = Peluang kejadian pada X, diakibatkan karena kejadian pada Y terjadi lebih dulu disebut juga probabilitas posterior
P(X) = Peluang kejadian pada X atau probabilitas sebelumnya
P(Y/X) = akibat setelah kejadian pada X, terdapat gejala tambahan dari kejadian Y,

a. Pendekatan Bayesian diperoleh atas dasar pada probabilitas subjektif; untuk setiap proposisi, probabilitas subjektivitas yang ditetapkan.
b. Jika E merupakan jumlah total semua informasi yang terdapat dalam sistem dalam suatu kejadian. Maka proposisi (P) menggambarkan semua kejadian dari E yang memiliki hubungan dengan suatu nilai yang merepresentasikan probabilitas. Merupakan turunan dari inferensi Bayesian.
c. Cara untuk menghitung probabilitas kejadian khusus dari suatu pengamatan disediakan dari Teorema Bayes.
d. Untuk poin yang utama adalah bukanlah cara menilai ini dibuat menjadi turun tetapi

bagaimana kita mengetahui atau bagaimana membuat kesimpulan dari proposisi menjadi nilai tunggal

Contoh:
T: Di sebuah pabrik ada dua mesin yang memproduksi baut. Mesin pertama memproduksi 75% baut dan mesin kedua memproduksi 25% sisanya. Dari mesin pertama 5% baut rusak dan dari mesin kedua 8% baut rusak. Sebuah baut dipilih secara acak, berapa peluang baut tersebut berasal dari mesin pertama, jika baut tersebut rusak?
Jawab :
Misalkan A kejadian baut rusak dan B kejadian baut berasal dari Mesin 1. Periksa apakah Anda dapat melihat dari mana probabilitas ini berasal!
P(B) = 0,75 P(B0) = 0,25 P(A|B) = 0,05 P(A|B0) = 0,08
Sekarang, gunakan Teorema Bayes untuk menemukan probabilitas yang diperlukan:
P(B|A) = P(A|B)P(B)
P(A|B)P(B) + P(A|B0)P(B0)
= 0,05 × 0,75
0,05 × 0,75 + 0,08 × 0,25
= 0,3846

5.2 Teori Certainty Factor

Teori probabilitas telah disebut oleh ahli matematika sebagai teori ketidakpastian yang dapat direproduksi Selain teori probabilitas subjek, teori alternatif adalah khusus dikembangkan untuk menangani kepercayaan manusia daripada interpretasi frekuensi klasik dari probabilitas Semua teori ini adalah contoh penalaran yang tidak tepat.
Teori Certainty and Trust

a. Certainy Factor (CF) adalah Teori dinyatakan dalam faktor kapastian.
b. CF yang menyatakan keyakinan terhadap suatu fakta (fakta, atau hipotesis) berdasarkan peristiwa atau pendapat ahli/pakar.
c. Dalam menghadapi ketidakpastian dalam sistem berbasis pengetahuan digunakan ada beberapa metode yang menggunakan CF.
d. Salah satu metode yang digunakan adalah angka 1 atau angka 100 untuk kebenaran mutlak (complete confidence) dan nilai 0 untuk error.
e. Probabilitas CF. Misalnya, ketika seseorang menyebutkan bahwa peluang untuk hujan mempunyai nilai 85%, maka akan hujan bernilai 85% atau tidak hujan bernilai 15%.
f. Dalam pendekatan probabilistik, dapat dikatakan CF untuk akan terjadinya turun hujan = 85, yang mempunyai arti kemungkinan besar akan terjadi turun hujan.

Faktor Certainty

a. Menjelaskan konsep certainty dan ketidakpastian.
b. CF ini merupakan konsep yang independent terhadap satu terhadap lainnya. CF ini tidak dapat digabungkan menurut probabilitas namun tetapi dapat digabungkan menurut formula berikut:

$$\mathbf{CF[P,E] = MB[P,E] - MD[P,E]} \quad \mathbf{(3)}$$

Dimana :
CF merupakan factor kepastian
MB merupakan nilai kepercayaan
MD merupakan nilai ketidakpercayaan
P Merupakan probabilitas
E merupakn kejadian

Mengkombinasikan CF

Beberapa faktor kepastian dalam satu aturan yang dikombinasikan
IF inflasi tinggi, CF = 40%, (A), AND
IF tarifnya di atas 80%, CF=80%, (B), AND
IF harga barang turun, CF=100%, C THEN harga saham turun

CF(A, B, dan C) = min[CF(A), CF(B), CF(C)] (4)
CF(A, B, atau C) = maks[CF(A), CF(B), CF(C)] (5)

Menggabungkan 2 atau lebih dari aturan
R1: IF tingkat inflasi kurang dari 5%
THEN harga barang di pasar akan naik (CF = 0,7)
R2 : IF kecepatan pergerakan kurang dari 7%,
THEN harga barang di pasar akan naik (CF = 0.6)
CF(R1,R2) = CF(R1) + CF(R2)[1 – CF(R1)]; OR (6)
CF(R1,R2)= CF(R1) + CF(R2) – CF(R1)xCF(R2) (7)

Dari kedua aturan di atas maka kita dapat menghitung CF(R1,R2) berikut ini :
CF(R1,R2) = 0,7 + 0,6(1- 0,7)
= 0,5 + 0,6 (0,3)
= 0,88
Bila terdapat 3 aturan yang dipakai maka:
CF(R1,R2,R3)=CF(R1,R2)+CF(R3)[1–CF(R1,R2)] (8)
Untuk metode yang sama, maka dapat ditentukan nilai dari CF untuk 4 aturan dan bisa lebih.

5.3 Teori Dempster Shafer

Dempster-Shafer (DST) merupakan sebuah teori matematika bukti. Karya mani pada subjek adalah, yang merupakan perluasan dari Dempster. Dalam ruang diskrit yang terbatas, teori Dempster-Shafer dapat diartikan sebagai generalisasi teori probabilitas di mana probabilitas ditetapkan ke himpunan yang bertentangan dengan sesuatu yang tunggal yang saling eksklusif. Dalam teori probabilitas tradisional, bukti dikaitkan dengan hanya satu kemungkinan peristiwa. Dalam DST, bukti dapat dikaitkan dengan beberapa kemungkinan peristiwa, misalnya, rangkaian peristiwa. Akibatnya, bukti dalam DST dapat bermakna pada tingkat abstraksi yang lebih tinggi tanpa harus menggunakan asumsi tentang peristiwa dalam set bukti. Dimana bukti cukup untuk memungkinkan penugasan probabilitas untuk peristiwa tunggal, model Dempster-Shafer runtuh ke formulasi probabilistik tradisional. Salah satu fitur yang paling penting dari teori Dempster-Shafer adalah bahwa model dirancang untuk mengatasi berbagai tingkat presisi mengenai informasi dan tidak ada asumsi lebih lanjut yang diperlukan untuk mewakili informasi. Hal ini juga memungkinkan untuk representasi langsung dari ketidakpastian tanggapan sistem di mana input yang tidak tepat dapat dicirikan oleh satu set atau interval dan output yang dihasilkan adalah satu set atau interval.
Ada tiga fungsi penting dalam teori Dempster-Shafer: fungsi penetapan probabilitas dasar (bpa atau m), fungsi Belief (Bel), dan fungsi Plausibility (Pl).
Penugasan probabilitas dasar (bpa) adalah primitif dari teori bukti. Secara umum, istilah "penugasan probabilitas dasar" tidak mengacu pada probabilitas dalam pengertian klasik. Bpa, diwakili oleh m, mendefinisikan pemetaan daya set ke interval antara 0 dan 1, di mana bpa dari himpunan nol adalah 0 dan penjumlahan dari bpa dari semua himpunan bagian dari himpunan daya adalah 1. Nilai dari bpa untuk himpunan A tertentu (diwakili sebagai m(A)), menyatakan proporsi dari semua bukti yang relevan dan tersedia yang mendukung klaim bahwa elemen tertentu dari X (kumpulan universal) termasuk dalam himpunan A tetapi tidak pada himpunan tertentu. himpunan bagian dari A Klir, 1998. Nilai m(A) hanya berkaitan dengan himpunan A dan tidak membuat klaim tambahan tentang himpunan bagian dari A. Bukti lebih lanjut tentang himpunan bagian A akan diwakili oleh bpa lain, yaitu B A, m(B) akan bpa untuk subset B. Secara formal, deskripsi m ini dapat direpresentasikan dengan tiga persamaan berikut:

m: P(X) [0,1] (9)

$m(\emptyset) = 0$ (10)

$$\sum_{A \epsilon P(X)} m(A) = 1 \quad (11)$$

di mana P (X) mewakili himpunan daya X, ∅ adalah himpunan nol, dan A adalah himpunan dalam himpunan daya (Aϵ P (X)).

Beberapa peneliti telah menemukan itu berguna untuk menafsirkan tugas probabilitas dasar sebagai probabilitas klasik, seperti Chokr dan Kreinovich, 1994, dan kerangka teori Dempster-Shafer dapat mendukung interpretasi ini. Implikasi teoritis dari interpretasi ini dikembangkan dengan baik di [Kramosil, 2001]. Ini adalah interpretasi yang sangat penting dan berguna dari teori Dempster-Shafer tetapi tidak menunjukkan cakupan penuh dari kekuatan representasi dari tugas probabilitas dasar. Dengan demikian, bpa tidak dapat disamakan dengan probabilitas klasik secara umum. Dari penetapan probabilitas dasar, batas atas dan bawah suatu interval dapat ditentukan. Interval ini berisi probabilitas yang tepat dari serangkaian minat (dalam pengertian klasik) dan dibatasi oleh dua ukuran kontinu non-aditif yang disebut Belief dan Plausibility. Keyakinan batas bawah untuk himpunan A didefinisikan sebagai jumlah dari semua penetapan probabilitas dasar dari himpunan bagian yang tepat (B) dari himpunan yang diminati (A) (B A). Batas atas, Plausibility, adalah jumlah dari semua penetapan probabilitas dasar dari himpunan (B) yang memotong himpunan bunga (A) (B A). Secara formal, untuk semua himpunan A yang merupakan elemen himpunan pangkat (Aϵ P (X)).

$$Bel(A) = \sum_{B|B \subseteq A} m(B) \quad (12)$$

$$Pl(A) = \sum_{B|B \subseteq A} m(B) \quad (13)$$

Kedua ukuran, Belief dan Plausibility adalah nonaditif. Hal ini dapat diinterpretasikan sebagai tidak diperlukannya jumlah semua ukuran Keyakinan menjadi 1 dan demikian pula untuk jumlah ukuran Plausibility.

Dimungkinkan untuk mendapatkan penetapan probabilitas dasar dari ukuran Belief dengan fungsi invers berikut:

$$m(A) = \sum_{B|B \subseteq A} (-1)^{|A-B|} Bel(B) \quad (14)$$

dimana |A-B| adalah selisih kardinalitas kedua himpunan.

Selain menurunkan ukuran-ukuran ini dari penetapan probabilitas dasar (m), kedua ukuran ini dapat diturunkan satu sama lain. Misalnya, Plausibility dapat diturunkan dari Belief dengan cara berikut:

$$Pl\ (A) = 1 - Bel\ (\bar{A}) \quad (15)$$

di mana A adalah komplemen klasik dari A. Definisi Plausibility dalam hal Keyakinan ini berasal dari fakta bahwa semua tugas dasar harus berjumlah 1.

$$m(\bar{A}) = \sum_{B|B \subseteq \bar{A}} m(B) = \sum_{B|B \cap A = \emptyset} m(B) \quad (16)$$

$$\sum_{B|B \cap A \neq \emptyset} m(B) = \sum_{B|B \cap A = \emptyset} m(B) \quad (17)$$

Dari definisi Belief dan Plausibility, maka Pl(A) 1– Bel(A) . Sebagai konsekuensi dari Persamaan 6 dan 7, diberikan salah satu dari ukuran ini (m(A), Bel(A), Pl(A)) adalah mungkin untuk menurunkan nilai dari dua ukuran lainnya. Probabilitas yang tepat dari suatu peristiwa (dalam pengertian klasik) masing-masing terletak di dalam batas bawah dan atas Keyakinan dan Masuk Akal.

$$Bel(A) = P(A) = Pl\ (A) \quad (18)$$

Probabilitas ditentukan secara unik jika Bel (A) = Pl(A). Dalam hal ini, yang sesuai dengan probabilitas klasik, semua probabilitas, P(A) ditentukan secara unik untuk semua himpunan bagian A dari himpunan universal X. Jika tidak, Bel (A) dan Pl(A) masing-masing dapat dilihat sebagai batas bawah dan atas pada probabilitas, di mana probabilitas aktual terkandung dalam interval yang dijelaskan oleh batas. Probabilitas atas dan bawah yang diturunkan oleh kerangka kerja lain dalam teori informasi umum tidak dapat secara langsung ditafsirkan sebagai fungsi Belief dan Plausibility.

PETUNJUK ATURAN KOMBINASI

Tujuan dari agregasi informasi adalah untuk meringkas dan menyederhanakan kumpulan data secara bermakna apakah data tersebut berasal dari satu sumber atau banyak sumber. Contoh akrab teknik agregasi termasuk rata-rata aritmatika, rata-rata geometris, rata-rata harmonik, nilai maksimum, dan nilai minimum. Aturan kombinasi adalah jenis khusus dari metode agregasi untuk data yang diperoleh dari berbagai sumber. Berbagai sumber ini memberikan penilaian yang berbeda untuk kerangka penegasan yang sama dan teori Dempster-Shafer didasarkan pada asumsi bahwa sumber-sumber ini independen. Persyaratan untuk menetapkan independensi sumber merupakan pertanyaan filosofis yang penting.

Dari sudut pandang teoretis himpunan, aturan-aturan ini berpotensi menempati kontinum antara konjungsi (AND-berbasis persimpangan himpunan) dan disjungsi (OR-berbasis himpunan serikat). Dalam situasi di mana semua sumber dianggap dapat diandalkan, operasi konjungtif adalah tepat (A dan B dan C…). Dalam kasus di mana ada satu sumber terpercaya di antara banyak sumber, kita dapat membenarkan penggunaan operasi kombinasi disjungtif (A atau B atau C…). Namun, banyak operasi kombinasi terletak di antara dua ekstrem ini (A dan B atau C, A dan C atau B, dll.). Dubois dan Prade menggambarkan ketiga jenis kombinasi ini sebagai conjunctive pooling ($A \cap B$, jika $A \cap B \neq \emptyset$), disjunctive pooling ($A \cup B$), dan tradeoff (Ada banyak cara tradeoff antara $A \cap B$ dan $A \cup B$ dapat dicapai).

Ada beberapa operator yang tersedia di setiap kategori penyatuan yang dengannya kumpulan data dapat digabungkan. Salah satu cara perbandingan aturan kombinasi adalah dengan membandingkan sifat aljabar yang dipenuhinya. Dengan jenis operasi kombinasi tradeoff, lebih sedikit informasi yang diasumsikan daripada dalam pendekatan Bayesian dan sebagai konsekuensinya presisi hasil dapat terganggu. Di sisi lain, jawaban tepat yang diperoleh melalui pendekatan Bayesian tidak mengungkapkan ketidakpastian apa pun yang terkait dengannya dan mungkin memiliki asumsi tersembunyi tentang aditif atau Prinsip Alasan yang Tidak Cukup.

Sesuai dengan gagasan umum tentang kontinum operasi kombinasi ini, ada beberapa kemungkinan cara di mana bukti dapat digabungkan dalam teori Dempster-Shafer. Aturan kombinasi asli dari beberapa penugasan probabilitas dasar yang dikenal sebagai aturan Dempster adalah generalisasi dari aturan Bayes. Aturan ini sangat menekankan kesepakatan antara berbagai sumber dan mengabaikan semua bukti yang saling bertentangan melalui faktor normalisasi. Ini dapat dianggap sebagai operasi DAN yang ketat. Penggunaan aturan Dempster mendapat kecaman serius ketika terjadi konflik yang signifikan dalam informasi. Akibatnya, peneliti lain telah mengembangkan aturan Dempster yang dimodifikasi yang mencoba untuk mewakili tingkat konflik dalam hasil akhir. Masalah konflik ini dan alokasi massa bpa yang terkait dengannya adalah perbedaan kritis antara semua aturan tipe Dempster. Untuk menggunakan salah satu aturan kombinasi ini dalam aplikasi, penting untuk memahami bagaimana konflik harus diperlakukan dalam konteks aplikasi tertentu.

Selain aturan kombinasi Dempster, kita akan membahas empat aturan Dempster yang dimodifikasi: Aturan Yager; Aturan kombinasi terpadu Inagaki; aturan kombinasi tengah Zhang; dan aturan penyatuan disjungtif Dubois dan Prade. Tiga jenis rata-rata akan dipertimbangkan: diskon dan kombinasi; rata-rata konvolutif; dan pencampuran. Semua aturan kombinasi akan dianggap relatif terhadap empat sifat aljabar: komutatif, A * B = B * A; idempotensi, A * A = A; kontinuitas, A * B = A' * B, di mana A' = A (A'sangat dekat dengan A); dan asosiatif, A * (B * C) = (A * B) * C; di mana * menunjukkan operasi kombinasi. Motivasi untuk sifat-sifat ini dibahas panjang lebar.

ATURAN KOMBINASI DEMPSTER

Aturan kombinasi Dempster sangat penting untuk konsepsi asli teori Dempster-Shafer. Ukuran Belief dan Plausibility berasal dari tugas dasar gabungan. Aturan Dempster menggabungkan beberapa fungsi keyakinan melalui penetapan probabilitas dasar (m). Fungsi keyakinan ini didefinisikan pada kerangka penegasan yang sama, tetapi didasarkan pada argumen independen atau kumpulan bukti. Isu independensi merupakan faktor kritis ketika menggabungkan bukti dan merupakan subjek penelitian penting dalam teori Dempster-Shafer. Aturan kombinasi Dempster adalah murni operasi konjungtif (AND). Aturan kombinasi menghasilkan fungsi keyakinan berdasarkan bukti gabungan konjungtif.

Secara khusus, kombinasi (disebut joint m12) dihitung dari agregasi dua m1 dan m2 bpa dengan cara berikut:

$$m_{12}(A) = \frac{\sum_{B \cap C = A} m1(B) m2(C)}{1-K}, \text{ Ketika A} \neq \emptyset \quad (19)$$

$$m_{12}(\emptyset) = 0$$

$$\text{Dimana K} = \sum_{B \cap C = \emptyset} m_1(B) m_2(C) \quad (20)$$

K mewakili massa probabilitas dasar yang terkait dengan konflik. Ini ditentukan dengan menjumlahkan hasil kali bpa dari semua himpunan di mana perpotongannya nol. Aturan ini komutatif, asosiatif, tetapi tidak idempoten atau kontinu.

Penyebut dalam aturan Dempster, 1-K, adalah faktor normalisasi. Ini memiliki efek sepenuhnya mengabaikan konflik dan menghubungkan setiap massa probabilitas yang terkait dengan konflik ke himpunan nol. Akibatnya, operasi ini akan menghasilkan hasil yang berlawanan dalam menghadapi konflik yang signifikan dalam konteks tertentu. Masalah dengan bukti yang bertentangan dan aturan Dempster awalnya ditunjukkan oleh Lotfi Zadeh dalam ulasannya tentang buku Shafer, A Mathematical Theory of Evidence. Zadeh memberikan contoh yang meyakinkan tentang hasil yang salah. Misalkan seorang pasien diperiksa oleh dua dokter mengenai gejala neurologis pasien. Dokter pertama percaya bahwa pasien memiliki meningitis dengan probabilitas 0,99 atau tumor otak, dengan probabilitas 0,01. Dokter kedua percaya pasien benar-benar menderita gegar otak dengan probabilitas 0,99 tetapi mengakui kemungkinan tumor otak dengan probabilitas 0,01. Menggunakan nilai untuk menghitung m (tumor otak) dengan aturan Dempster, kami menemukan bahwa m(tumor otak) = Bel (tumor otak) = 1. Jelas, aturan kombinasi ini menghasilkan hasil yang menyiratkan dukungan lengkap untuk diagnosis bahwa keduanya dokter dianggap sangat tidak mungkin.

Mengingat contoh sederhana namun dramatis dari hasil berlawanan dari faktor normalisasi dalam aturan Dempster, sejumlah metode dan operasi kombinasi yang telah dikembangkan untuk mengatasi masalah ini diajukan oleh bukti yang sangat bertentangan. Kami akan membahas banyak dari alternatif ini di bagian berikut serta pentingnya konflik dan konteks dalam pemilihan aturan. Kami akan menemukan bahwa selain tingkat atau derajat konflik penting dalam menentukan kepatutan penggunaan aturan Dempster, relevansi konflik juga memainkan peran penting.

METODE DISKON + KOMBINASI

Metode tradeoff ini awalnya dibahas dalam Shafer, 1976 dan menangani konflik hanya dengan cara yang tersirat dari namanya. Secara khusus, ketika seorang analis dihadapkan dengan bukti yang bertentangan, dia dapat mendiskontokan sumbernya terlebih dahulu, dan kemudian menggabungkan fungsi yang dihasilkan dengan aturan Dempster (atau aturan alternatif) menggunakan fungsi diskon. Fungsi pendiskontoan ini harus memperhitungkan keandalan mutlak sumber. Keandalan absolut menyiratkan

bahwa analis memenuhi syarat untuk membuat perbedaan antara keandalan ahli, sensor, atau sumber informasi lainnya dan dapat mengungkapkan perbedaan antara sumber ini secara matematis.
Shafer menerapkan fungsi diskon untuk setiap Keyakinan yang ditentukan. Biarkan 1-□i menjadi tingkat keandalan yang dapat diatribusikan ke fungsi kepercayaan tertentu, A (Shafer menyebutnya sebagai tingkat kepercayaan), di mana $0 \leq \alpha i \leq 1$ dan i adalah indeks yang digunakan untuk menentukan fungsi diskonto tertentu yang terkait dengan a ukuran keyakinan tertentu. Bel α^i (A) kemudian mewakili fungsi kepercayaan yang didiskon yang didefinisikan oleh:

$$Bel^{\alpha_i}(A) = (1 - \alpha_i) Bel(A) \quad (21)$$

Shafer kemudian merata-ratakan semua fungsi kepercayaan yang terkait dengan himpunan A (Bel $\alpha^i{}_1$(A) Bel $\alpha^i{}_2$(A), …. Bel α^i n(A)) untuk mendapatkan rata-rata n Bel, dilambangkan dengan Bel .

$$Bel(A) = \frac{1}{n}(Bel^{\alpha}1(A) + \cdots + Bel^{\alpha}n(A)) \quad (22)$$

untuk semua himpunan bagian A dari himpunan semesta X.
Akibatnya, metode diskon dan kombinasi menggunakan fungsi rata-rata sebagai metode kombinasi. Ini digunakan ketika semua fungsi keyakinan yang akan digabungkan sangat bertentangan dan tingkat diskonto tidak terlalu kecil. Ini juga dapat digunakan untuk menghilangkan pengaruh dari setiap fungsi keyakinan tunggal yang sangat bertentangan asalkan fungsi keyakinan yang tersisa tidak terlalu bertentangan satu sama lain dan tingkat diskonto tidak terlalu kecil atau terlalu besar. Sebagai alternatif, untuk kasus ini orang juga bisa menghilangkan keyakinan yang sangat bertentangan sama sekali jika itu masuk akal.

Data diberikan oleh nilai diskrit
Misalkan dua ahli berkonsultasi mengenai kegagalan sistem. Kegagalan dapat disebabkan oleh Komponen A, Komponen B atau Komponen C. Ahli pertama berpendapat bahwa kegagalan disebabkan oleh Komponen A dengan probabilitas 0,99 atau Komponen B dengan probabilitas 0,01 (dilambangkan dengan m1(A) dan m1(B), masing-masing). Ahli kedua berpendapat bahwa kegagalan tersebut disebabkan oleh Komponen C dengan probabilitas 0,99 atau Komponen B dengan probabilitas 0,01 (masing-masing dilambangkan dengan m2(C) dan m2(B). Distribusi dapat diwakili oleh yang berikut:
Ahli 1:
m1(A) = 0,99 (gagal karena Komponen A)
m1(B) = 0,01 (gagal karena Komponen B)

Ahli 2:
m2(B) = 0,01 (gagal karena Komponen B)
m2(C) = 0,99 (gagal karena Komponen C)

Kombinasi massa yang terkait dengan para ahli dirangkum dalam Tabel 5.1.
Tabel 5.1: Kombinasi Dempster Expert 1 dan Expert 2

			Expert 1			
			A	B	C	Failure Cause
			0.99	0.01	0	m1
Expert 2	Failure Cause	m2				

	A	0	m1(A) m2(A) = 0	m1(B) m2(A) = 0	m1(C) m2(A) = 0	
	B	0.01	m1(A) m2(B) = 0.0099	m1(B) m2(B) = 0.0001	m1(C) m2(B) = 0	
	C	0.99	m1(A) m2(C) = 0.9801	m1(B) m2(C) = 0.0099	m1(C) m2(C) = 0	

Menggunakan Persamaan 19-21:

1. Untuk menghitung penetapan probabilitas dasar gabungan untuk sel tertentu, cukup kalikan massa dari kolom dan baris terkait.
2. Bila perpotongan tidak kosong, massa untuk himpunan tertentu dari setiap sumber dikalikan, misalnya, m12(B) = (0,01)(0,01) = 0,0001.
3. Dimana persimpangan kosong, ini merupakan bukti yang bertentangan dan harus dihitung juga. Untuk perpotongan kosong dari dua himpunan A dan C yang masing-masing berhubungan dengan Pakar 1 dan 2, ada massa yang terkait dengannya. m1(A) m2(C)=(0.99)(0.99) =(0.9801).
4. Kemudian jumlahkan massa untuk semua himpunan dan konfliknya.
5. Satu-satunya nilai bukan nol adalah untuk kombinasi B, m12(B) = 0,0001. Dalam contoh ini hanya ada satu persimpangan yang menghasilkan B, tetapi dalam contoh yang lebih rumit dimungkinkan untuk menemukan lebih banyak persimpangan untuk menghasilkan B.
6. Untuk K, ada tiga sel yang berkontribusi terhadap konflik yang diwakili oleh persimpangan kosong. Menggunakan Persamaan 13, K = (0,99)(0,01) + (0,99)(0,01) + (0,99)(0,99) = 0,9999
7. Menggunakan Persamaan 11, hitung sambungannya, m1(B) m2(B) = (.01)(.01) / [1-0.9999] = 1

Meskipun ada bukti yang sangat bertentangan, penetapan probabilitas dasar untuk kegagalan Komponen B adalah 1, yang sesuai dengan Bel (B) = 1. Ini adalah hasil dari normalisasi massa untuk mengecualikan yang terkait dengan konflik. Ini menunjukkan inkonsistensi ketika aturan Dempster digunakan dalam situasi konflik relevan yang signifikan yang ditunjukkan oleh Zadeh.

Contoh :
Seorang perempuan bernama Nanik, beliau sedang mengalami gejala panas badan di seluruh tubuhnya. Dokter mengdiagnosa, penyakit yang kelihatannya diderita oleh nanik adalah sakit flu , sakit demam, atau sakit bronchitis.
Gejala 1
Setelah dilakukan observasi terhadapa nanik, diketahui nilai kepercayaan sebagai gejala dari sakit flu, sakit demam, dan sakit bronchitis adalah :
M1 {Fl,De,Bu} = 0,8
M1{θ} = 1 – 0,8 = 0,2
Kemudia dihari berikutnya Nanik datang lagi dan membawa gejala yang baru yaitu hidungnya mengalami buntu.
Gejala 2
Untuk gejala alergi, penyakit flu, dan demam terhadap nilai kepercayaan setelah dilakukan observasi terhadap hidung buntu yaitu
M2{Al,Fl,De} = 0,9
M2{θ) = 1 – 0,9 = 0,1

$$m_3(Z) = \frac{\sum_{x \cap y = Z} m1(x)m2(y)}{1 - \sum x \cap y = \emptyset^{m1(x)m2(y)}}$$

Tabel meunjukkan aturan kombinasi tersebut

	{Al,Fl,De} (0,9)	**θ (0,1)**
{Fl,De,Bu} (0,8)	{Fl,De) (0,72)	(Fl,De,Bu) {0.08)
Θ (0,2)	{Al,Fl.De} (0,18)	θ (0,02)

Dapat dihitung:

$$m_3(Fl, De) = \frac{0{,}72}{1 - 0} = 0{,}72$$

$$m_3(\emptyset) = \frac{0{,}02}{1 - 0} = 0{,}02$$

$$m_3(Fl, De, Bu) = \frac{0{,}08}{1 - 0} = 0{,}08$$

Dari sini kita dapat melihat bahwa, pada awalnya hanya ada panas, m(Fl,De,Bu)= 0,8; tetapi setelah ada gejala baru yaitu hidung tersumbat maka nilai m{Fl,De,Bu} = 0,08. Demikian pula, pada awalnya hanya ada gejala hidung tersumbat m{Al,Fl,De} = 0,9; namun setelah ada gejala baru yaitu panas, maka nilai m{Al,Fl,De} = 0.18. Dengan 2 gejala tersebut nilai densitas terkuat adalah m{Fl,De} yaitu 0,72.
gejala 3
Keesokan harinya, Nanik datang lagi, dan pesta minggu lalu dia baru saja piknik.
Jika nilai kepercayaan diketahui setelah mengamati piknik sebagai gejala alergi, yaitu:
M4{Al} =0,6
M4{θ) = 1 - 0,6 = 0,4
Kemudian kita dapat mencari aturan kombinasi dengan nilai kepercayaan m5 seperti yang ditunjukkan pada table

		{Al}	**(0,6)**	**θ**	**(0,4)**
{Fl,De}	(0,72)	Ø	(0,432)	{Fl,De}	(0,228)
{Al,Fl,De}	(0,18)	{Al}	(0,108)	{Al,Fl,De}	(0,072)
{Fl,De,Bu}	(0,08)	Ø	(0,048)	{Fl,De,Bu}	(0,032)
{θ}	(0,02)	{Al}	(0,012)	θ	(0,008)

Sehingga dapat dihitung :

$m_5(Al) = \frac{0,108+0,012}{1-(0,432+0,048)} = 0,231$

$$m_5(Fl, De) = \frac{0,288}{1-(0,432+0,048)} = 0,554$$

$$m_5(Al, Fl, De) = \frac{0,0722}{1-(0,432+0,048)} = 0,138$$

$$m_5(Fl, De, Bu) = \frac{0,032}{1-(0,432+0,048)} = 0,062$$

$$m_5(\emptyset) = \frac{0,008}{1-(0,432+0,048)} = 0,015$$

Densitas baru m_5 adalah sbb :

$$m_5\{Al\} = \frac{0,108+0012}{1-(0,432+0,048)} = 0,231$$

$$m_5\{Fl, De\} = \frac{0,288}{1-(0,432+0,048)} = 0,554$$

$$m_5\{Al, Fl, De\} = \frac{0,072}{1-(0,432+0,048)} = 0,138$$

$$m_5\{Fl, De, Bu\} = \frac{0,032}{1-(0,432+0,048)} = 0,062$$

$$m_5\{\theta\} = \frac{0,008}{1-(0,432+0,048)} = 0,015$$

Ternyata dengan gejala baru ini karena nanikterhadap alergi terhadap udara, nilai densitas yang paling tetap yaitu m_5{Fl,De} = 0,554.
Jadi dengan tiga jenis gejala yang dialami oleh Nanik, kemungkinan paling kuat Nanik terkena Flue dan Demam.

Soal :
Bayu merupakan calon mahasiswa Fakultas Sains dan Teknologi Umsida berasal dari kota Pasuruan. Terdapat 3 jurusan yang diminati oleh Bayu yaitu Teknik Mesin (M), Elektro (E) dan Teknik Sipil(S). Untuk itu dia mencoba mengikuti beberapa test uji coba. Ujicoba pertama test Logika dengan hasil test menunjukkan bahwa probabilitas densitas m_1{M,E} = 0,75.
Test kedua adalah test matematika, hasil test menunjukkan bahwa probabilitas densitas m_2(M} = 0,8.
Test ketiga adalah wawancara. Hasil test menunjukkan bahwa densitas probabilitas m3{S} = 0,3.
Tentukan probabilitas densitas dari kombinasi Test (hasil test) yang didapat oleh Bayu.

BAB 6
Sistem Pakar

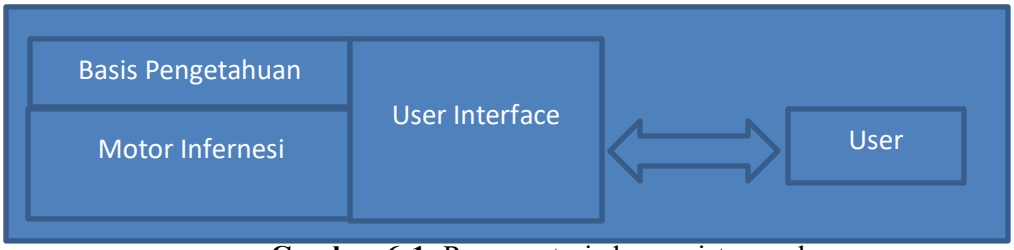
6.1 Pendahuluan

Kecerdasan buatan (AI) berkaitan dengan merancang sistem komputer cerdas yang mengubah sesuatu dari berguna menjadi sesuatu yang penting. Salah satu bidang AI yang dapat mengklaim tanggung jawabbesar atas kesadaran AI yang meningkat saat ini adalah Sistem Pakar Berbasis Pengetahuan, program komputer yang mewujudkan keahlian manusia. Ini adalah salah satu bidang penelitian penting dalam Kecerdasan Buatan untuk pemecahan masalah. Ini adalah teknologi AI pertama yang berdampak luas pada bisnis dan industri. Ini menawarkan banyak penggunaan praktis dan potensi komersial. Ada banyak aplikasi yang memiliki tugas pemecahan masalah komprehensif yang bersifat algoritmik atau deterministik, seperti mengevaluasi, menguji, menjadwalkan, dan menghitung serta menyajikan grafik.

Namun demikian, teori di balik proses pengambilan keputusan dan praktik desain seperti sintesis, penilaian, dan simulasi proses pengambilan keputusan tidak terstruktur dengan baik dan melibatkan penilaian dan heuristik berbasis solusi. Sistem pakar membantu dalam aktivitas pemecahan masalah yang salah urus dan non-deterministik seperti dimensi signifikan pada siklus desain dan pengambilan keputusan. Shellyang efisien merupakan prasyarat dalam mengembangkan sistem pakar. Shell akan memiliki fasilitas untuk implementasi informasi yang mudah, dan untuk interaksi yang lancar antara pengguna dan jaringan pakar. Unit inti shell harus mencakup modul untuk akuisisi informasi, mekanisme inferensi dengan antarmukapengguna yang benar.

Sistem Pakar adalah program komputer yang dirancang untuk bertindak sebagai pakar dalam bidang atau domain keahlian tertentu. Sering disebut sebagai sistem pakar berbasis informasi, sistem pakar biasanya mencakup basis pengetahuan yang kuat yang terdiri dari fakta domain dan heuristik, atau pedoman untuk menerapkan fakta-fakta. Sistem pakar telah terbukti bermanfaat dalam memecahkan berbagai masalah lapangan, termasuk diagnosis medis,analisis kimia, pengujian geologi, modifikasi perangkat komputer, dan masalah teknologi lainnya.

Komponen kunci dari banyak sistem pakar adalah sebagai berikut dan diilustrasikan seperti pada gambar 6.1.



Gambar 6.1. Representasi skema sistem pakar

Berbagai definisi sistem pakar telah ditawarkan oleh beberapa penulis:

- Sistem pakar termasuk dalam bidang kecerdasan buatan, dan merupakan program komputer yang mensimulasikan penilaian dan perilaku individu yang memiliki pengetahuan dan

pengalaman ahli dalam bidang tertentu. Ini adalah program komputer berbasis pengetahuan yang menunjukkan tingkat keahlian dalam domain tertentu sehingga memecahkan masalah atau membuat keputusan yang sebanding dengan seorang ahli manusia.

- Bisa juga disebut sebagai program AI yang mencapai kompetensi tingkat ahli dalam memecahkan masalah di area tugas dengan membawa kumpulan pengetahuan tentang tugas-tugas tertentu. Ini bisa disebut sebagai sistem berbasis pengetahuan atau pakar
- Jenis program aplikasi yang membuat keputusan atau memecahkan masalah dalam bidang tertentu dengan menggunakan pengetahuan dan aturan analitis yang ditentukan oleh para ahli di bidang tersebut.
- Sistem pakar merupakan cabang dari kecerdasan buatan, yang bertujuan untuk mengambil pengalaman spesialis manusia dan untuk mentransfer ke sistem komputer. Pengetahuan khusus disimpan di komputer, yang oleh sistem eksekusi (mesin inferensi) adalah penalaran dan memperoleh kesimpulan khusus untuk masalah tersebut.
- Sebuah program komputer yang menggunakan pengetahuan dan teknik penalaran untuk memecahkan masalah yang biasanya membutuhkan kemampuan ahli manusia. Perangkat lunak yang menerapkan penalaran seperti manusia yang melibatkan aturan dan heuristik untuk memecahkan masalah. Sistem Pakar (EPS) adalah sistem perangkat lunak, yang menemukan solusi berdasarkan pengetahuan pakar atau memberikan evaluasi dari pengetahuan yang diketahui. Contohnya adalah sistem untuk mendukung diagnosis medis atau untuk analisis data ilmiah.
- Adalah pengembangan tertentu dari Kecerdasan Buatan yang membantu memecahkan masalah atau membuat keputusan melalui penggunaan penyimpanan Informasi yang relevan (dikenal sebagai Basis Pengetahuan, dan diturunkan dari satu atau lebih pakar manusia), dan seperangkat teknik penalaran.
- Sistem berbasis kecerdasan buatan yang mengubah pengetahuan seorang ahli dalam subjek tertentu menjadi kode perangkat lunak. Kode ini dapat digabungkan dengan kode lain yang sejenis (berdasarkan pengetahuan orang lain) dan digunakan untuk menjawab pertanyaan (queries) yang diajukan melalui komputer.
- Terlepas dari definisi Sistem Pakar ini oleh penulis yang berbeda, itu adalah Penting untuk meringkas bahwa sistem pakar adalah program komputer berbasis pengetahuan yang menunjukkan, dalam domain tertentu, tingkat keahlian dalam pemecahan masalah yang sebanding dengan seorang ahli manusia. Metode masalah ini menggunakan basis pengetahuan, yang dirumuskan secara hati-hati berdasarkan penilaian ahli, intuisi, dan pengalaman. Dengan demikian sistem pakar mewujudkan kognisi dan kemampuan seorang pakar dalam bidang tertentu, sehingga meniru kemampuan pengambilan keputusan manusia.

Istilah sistem pakar dicadangkan untuk program yang basis pengetahuannya berisi pengetahuan yang digunakan oleh pakar manusia, berbeda dengan pengetahuan yang dikumpulkan dari buku teks atau non-ahli. Kedua istilah, sistem pakar (ES) dan sistem berbasis pengetahuan (KBS), kadang-kadang digunakan secara sinonim. Secara bersama-sama, mereka mewakili jenis aplikasi AI yang paling luas. Bidang usaha intelektual manusia untuk ditangkap dalam sistem pakar disebut domain tugas. Tugas mengacu pada beberapa aktivitas pemecahan masalah yang berorientasi pada tujuan. Domain mengacu pada area di mana tugas sedang dilakukan. Tugas khas adalah diagnosis, perencanaan, penjadwalan, konfigurasi dan desain. Contoh domain tugas adalah penjadwalan awak pesawat. Dalam kebanyakan kasus, tujuan sistem pakar adalah untuk membantu dan mendukung penalaran pengguna dan bukan untuk menggantikan penilaian

pakar manusia. Faktanya, sistem pakar menawarkan solusi kepada pengguna yang tidak berpengalaman ketika pakar manusia tidak tersedia. Sistem pakar adalah bagian dari desain sistem cerdas di bawah AI, namun pakar dan kecerdasan bukanlah hal yang sama. Mari kita coba membedakan antara kecerdasan dan keahlian dengan maksud untuk mengungkap area fokus sistem pakar dalam diskusi luas tentang sistem cerdas.

6.1.1 Sistem pakar dan program konvensional

Sistem pakar berbeda dari program aplikasi tradisional dalam hal kemampuan mereka untuk menangani masalah dunia nyata yang menantang melalui proses aplikasi yang mencerminkan penilaian dan intuisi manusia.

Sistem pakar tidak harus bingung dengan program pemodelan kognitif, yang mencoba untuk mensimulasikan arsitektur mental manusia secara rinci. Sistem pakar adalah program praktis yang menggunakan strategi heuristik yang dikembangkan untuk memecahkan kelas masalah tertentu.

Tabel 6.1 Perbandingan aplikais pakar dengan konvensional

Aplikasi Sistem Pakar	Aplikasi Sistem Konvensional
Pengetahuan terfragmentasi dan implisit, sulit untuk dikomunikasikan kecuali dalam "potongan" kecil, dan sering didistribusikan di antara individu yang mungkin tidak setuju.	Pengetahuan lengkap dan eksplisit, dan mudah dikomunikasikan dengan rumus dan algoritma.
Aturan bersifat kompleks, bersyarat dan sering didefinisikan sebagai "aturan praktis" yang tidak tepat.	Aturannya sederhana dengan beberapa kondisi.
Sistem yang telah selesai menangkap, mendistribusikan, dan memanfaatkan keahlian	Produk jadi mengotomatiskan prosedur manual
Pemecahan masalah menuntut penerapan fakta, hubungan, dan aturan yang dinamis dan berdasarkan konteks	Pemecahan masalah membutuhkan urutan tindakan yang dapat diprediksi dan berulang.
Kinerja sistem diukur dalam derajat akurasi dan kelengkapan di mana penjelasan mungkin diperlukan untuk menetapkan kebenaran.	Kriteria sederhana digunakan untuk menentukan akurasi dan kelengkapan.
membuat kesimpulan	Alur program
Berbasis pengetahuan	Basis Data
Kelas objek	Kelas relasional

6.1.2 Peran Pengetahuan Heuristik dalam sistem pakar

Sebagian besar pengetahuan ahli domain dalam memecahkan masalah praktis terdiri dari heuristik yang diperoleh melalui pembelajaran dan pengalaman. Heuristik adalah aturan praktis, fakta, atau bahkan prosedur yang dapat digunakan untuk memecahkan beberapa masalah, tetapi tidak dijamin untuk melakukannya. Ini mungkin gagal. Heuristik dapat dengan mudah dianggap sebagai penyederhanaan deskripsi formal yang komprehensif dari sistem dunia nyata. Misalnya, dapat dibayangkan bahwa semua aspek pengoperasian mesin dapat sepenuhnya dijelaskan dalam model fisik atau matematika yang kompleks, termasuk keadaan di mana mesin tidak berfungsi. Pada prinsipnya, model ini dapat digunakan untuk menganalisis masalah mesin dan (secara algoritmik) menentukan malfungsi dengan kepastian virtual. Dalam prakteknya, model yang lengkap seringkali sulit untuk dikembangkan karena kurangnya informasi yang diperlukan tentang masalah dan kompleksitas yang melekat. Oleh karena itu, untuk banyak masalah, pakar domain merasa praktis dan perlu untuk menggantikan pengetahuan heuristik untuk model yang kompleks. Sistem sistem pakar mendapat manfaat dari prinsip heuristik.

6.2 Elemen Sistem Pakar

Sistem pakar menyimpan pengetahuan pakar dan menerapkannya "sesuai permintaan" untuk memecahkan masalah. Paling sering pengguna sistem pakar adalah orang. Pengguna juga dapat berupa sistem perangkat lunak lain atau bahkan perangkat mekanis. Seorang pengguna manusia, yang dikenal sebagai pengguna akhir biasanya memberikan informasi ke sistem pakar melalui terminal komputer. Sistem pakar menggunakan prosedur inferensi untuk menerapkan pengetahuan yang tersimpan pada fakta yang menggambarkan suatu masalah. Penerapan inferensi yang sistematis mengarah pada solusi yang kemudian ditampilkan di terminal.

Pengoperasian sistem pakar dapat dilihat dari interaksi komponen-komponen yang berbeda. Basis pengetahuan menyimpan pengetahuan tentang bagaimana memecahkan masalah. Prosedur inferensi dijalankan oleh modul perangkat lunak yang disebut mesin inferensi. Jika pengguna sistem pakar adalah seseorang, komunikasi dengan pengguna akhir ditangani melalui antarmuka pengguna akhir.

Masing-masing bagian utama dari arsitektur sistem pakar dapat dijelaskan lebih lanjut sebagai berikut:

6.2.1 Basis Pengetahuan

Pengetahuan disimpan dalam basis pengetahuan menggunakan simbol dan struktur data untuk mewakili konsep-konsep penting. Simbol dan struktur data dikatakan mewakili pengetahuan. Representasi pengetahuan dapat mengambil banyak bentuk. Bentuk yang paling umum adalah aturan produksi. Aturan produksi adalah cara yang sangat nyaman untuk mengekspresikan pengetahuan heuristik. Basis pengetahuan mengacu pada penyimpanan sebenarnya dari pengetahuan untuk sistem pakar tertentu.

Sebuah sistem representasi pengetahuan mungkin sederhana, hanya terdiri dari struktur data untuk mewakili aturan. Atau representasi pengetahuan dapat menggabungkan struktur lain yang lebih kompleks. Pengetahuan yang direpresentasikan dalam struktur data, seperti aturan, dikatakan dinyatakan secara deklaratif. Pengetahuan deklaratif adalah pengetahuan yang dinyatakan secara eksplisit dan dimaksudkan untuk dapat diakses oleh orang-orang yang mungkin perlu melihatnya, seperti pakar domain. Kemampuan untuk membuat pengetahuan deklaratifnya dapat diakses dan dimengerti adalah salah satu layanan terpenting yang disediakan oleh sistem representasi pengetahuan.

6.2.2 Mesin Inferensi

Mesin inferensi adalah modul perangkat lunak yang mengeksekusi prosedur untuk menerapkan pengetahuan untuk menghasilkan informasi baru tentang suatu masalah. Dalam sistem aturan produksi, mesin inferensi membandingkan aturan dengan fakta yang diketahui dalam file konteks untuk menentukan apakah fakta baru dapat disimpulkan. Kondisi dalam premis, atau bagian JIKA, dari aturan produksi dibandingkan dengan fakta yang diketahui. Jika kondisi ini terpenuhi, fakta-fakta dalam kesimpulan, atau MAKA bagian, dapat disimpulkan. Fakta-fakta yang baru disimpulkan kemudian ditambahkan ke file konteks sistem pakar.

6.2.3 Antarmuka Sistem Pakar

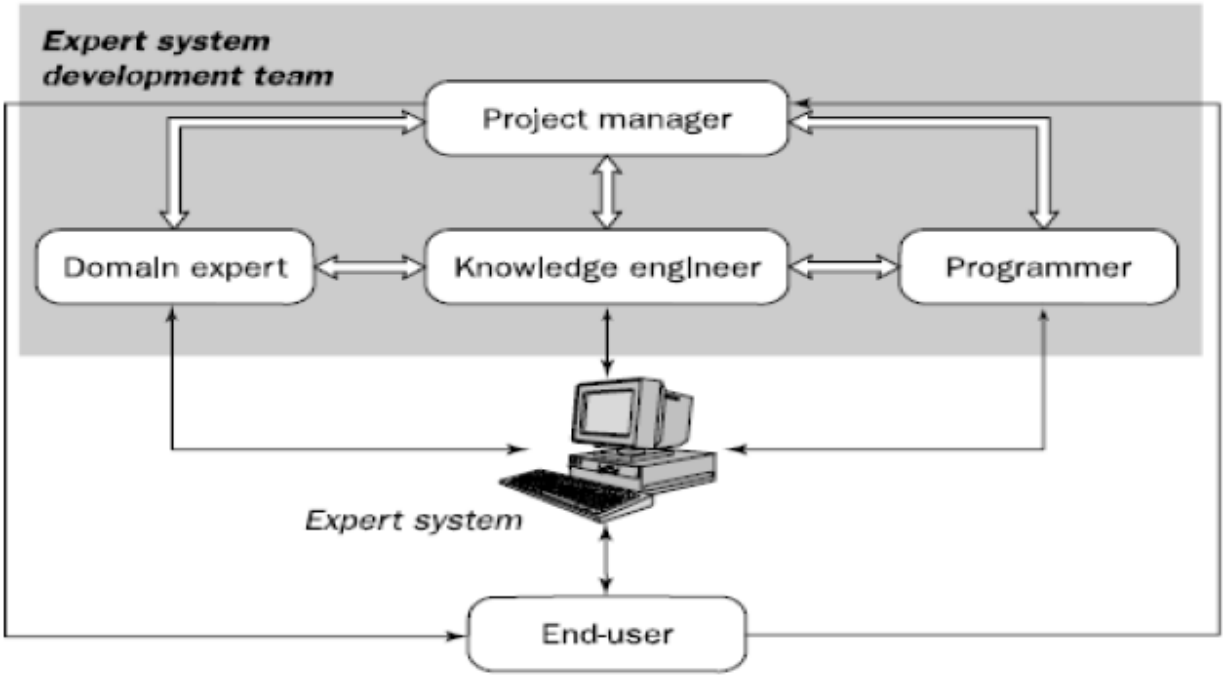
Sistem pakar berkomunikasi dengan pengguna manusia serta perangkat lunak dan sistem perangkat keras lainnya. Sistem pakar berkomunikasi dengan pengguna manusia melalui antarmuka pengguna akhir. Tujuan dari antarmuka pengguna akhir adalah untuk mendapatkan informasi tentang masalah dari pengguna akhir dan untuk menampilkan solusi. Untuk mendapatkan informasi, antarmuka dapat menampilkan pertanyaan di terminal dan meminta jawaban dari pengguna akhir. Solusi dapat terdiri dari pernyataan teks. Antarmuka pengguna akhir yang lebih rumit dapat menggunakan grafik dan hypertext. Fungsi yang berguna dari sistem pakar adalah kemampuan untuk menjelaskan tindakannya. Saat menggunakan sistem pakar, pengguna akhir mungkin ingin tahu mengapa pertanyaan diajukan atau mengapa fakta tertentu disimpulkan. Ketika solusi ditampilkan kepada pengguna akhir, pengguna dapat meminta penjelasan tentang bagaimana solusi itu

dicapai. Antarmuka pengguna akhir berisi prosedur yang menghasilkan penjelasan yang dapat ditampilkan kepada pengguna akhir. Dalam banyak aplikasi praktis, sistem pakar harus berinteraksi, dan bertukar data, dengan perangkat lunak dan sistem perangkat keras lainnya. Jumlah sistem pakar yang memiliki pengguna non-manusia, seperti sistem perangkat lunak lain atau perangkat kontrol proses elektronik, semakin meningkat. Oleh karena itu, penting untuk dicatat bahwa antarmuka sistem pakar dengan pengguna akhir dan komponen non-manusia lainnya.

6.3 Elemen Manusia dalam pengembangan Sistem Pakar

Membangun sistem pakar dikenal sebagai rekayasa pengetahuan dan praktisinya disebut insinyur pengetahuan. Insinyur pengetahuan harus memastikan bahwa komputer memiliki semua pengetahuan yang dibutuhkan untuk memecahkan masalah.

Selain itu, penting untuk mengidentifikasi berbagai jenis orang yang dibutuhkan untuk mengembangkan dan menggunakan sistem pakar dan keterampilan apa yang dibutuhkan. Secara umum, ada lima anggota tim pengembangan sistem pakar: pakar domain, insinyur pengetahuan, programmer, manajer proyek, dan pengguna akhir. Keberhasilan sistem pakar sepenuhnya tergantung pada seberapa baik anggota bekerja sama.



Gambar 6.2: Menampilkan faktor manusia dalam tim pengembangan sistem pakar

1. Pakar domain: Siapa pun dapat dianggap sebagai pakar domain jika dia memiliki pengetahuan yang mendalam (baik fakta maupun aturan) dan pengalaman praktis yang kuat dalam domain tertentu. Secara umum, seorang ahli adalah orang yang terampil yang dapat melakukan hal-hal yang tidak dapat dilakukan orang lain. Pertimbangkan beberapa contoh Pakar Manusia: Seorang dokter, grand master Catur, Penyihir keuangan, Koki, Insinyur, arsitek, apoteker, dll.
2. Insinyur pengetahuan: Individu yang mengkodekan pengetahuan pakar dalam bentuk deklaratif yang dapat digunakan oleh sistem pakar. Mereka juga orang-orang yang mampu

merancang, membangun dan menguji sistem pakar. Orang ini bertanggung jawab untuk memilih tugas yang sesuai untuk sistem pakar. Dia mewawancarai pakar domain untuk mengetahui bagaimana masalah tertentu diselesaikan. Melalui interaksi dengan pakar, perekayasa pengetahuan menetapkan metode penalaran apa yang digunakan pakar untuk menangani fakta dan aturan dan memutuskan bagaimana merepresentasikannya dalam sistem pakar. Insinyur pengetahuan kemudian memilih beberapa perangkat lunak pengembangan atau cangkang sistem pakar, atau melihat bahasa pemrograman untuk mengkodekan pengetahuan (dan terkadang mengkodekannya sendiri). Dan akhirnya, insinyur pengetahuan bertanggung jawab untuk menguji, merevisi, dan mengintegrasikan sistem pakar ke tempat kerja. Dengan demikian, insinyur pengetahuan berkomitmen untuk proyek dari tahap desain awal hingga pengiriman akhir sistem, dan bahkan setelah proyek selesai, dia mungkin juga terlibat dalam pemeliharaan sistem.

3. Pemrogram: adalah orang yang bertanggung jawab atas pemrograman yang sebenarnya, menggambarkan pengetahuan domain dalam istilah yang dapat dipahami oleh komputer. Pemrogram perlu memiliki keterampilan dalam pemrograman simbolik dalam bahasa AI seperti LISP, dan Prolog dan juga beberapa pengalaman dalam penerapan berbagai jenis cangkang sistem pakar. Selain itu, programmer harus mengetahui bahasa pemrograman konvensional seperti Python, R, Java, Julia, C, Pascal, FORTRAN dll. Jika shell sistem pakar digunakan, insinyur pengetahuan dapat dengan mudah mengkodekan pengetahuan ke dalam sistem pakar dan dengan demikian menghilangkan kebutuhan programmer. Namun, jika shell tidak dapat digunakan, seorang programmer harus mengembangkan pengetahuan dan struktur representasi data (basis pengetahuan dan database), struktur kontrol (mesin inferensi) dan struktur dialog (antarmuka pengguna). Pemrogram juga dapat terlibat dalam pengujian sistem pakar.
4. Manajer proyek: adalah pemimpin tim pengembangan sistem pakar, yang bertanggung jawab untuk menjaga proyek tetap pada jalurnya. Dia memastikan bahwa semua hasil dan pencapaian terpenuhi, berinteraksi dengan pakar, insinyur pengetahuan, pemrogram, dan pengguna akhir.
5. Pengguna: sering disebut hanya pengguna akhir, adalah individu yang akan berkonsultasi dengan sistem untuk mendapatkan saran yang akan diberikan oleh pakar. Dia adalah orang yang menggunakan sistem pakar ketika dikembangkan. Penggunanya mungkin ahli kimia analitik yang menentukan struktur molekul tanah dari Mars, dokter junior yang mendiagnosis penyakit darah menular, ahli geologi eksplorasi yang mencoba menemukan deposit mineral baru, atau operator sistem tenaga yang membutuhkan saran dalam keadaan darurat. Masing-masing pengguna sistem pakar ini memiliki kebutuhan yang berbeda, yang harus dipenuhi oleh sistem: penerimaan akhir sistem akan bergantung pada kepuasan pengguna. Pengguna tidak hanya harus percaya diri dengan kinerja sistem pakar tetapi juga merasa nyaman menggunakannya. Oleh karena itu, desain antarmuka pengguna sistem pakar juga penting untuk keberhasilan proyek; kontribusi pengguna akhir di sini bisa sangat penting.

   Kemungkinan Kelas Pengguna

   - Klien non-ahli yang mencari nasihat langsung - ES bertindak sebagai Konsultan atau Penasihat
   - Seorang siswa yang ingin belajar
   - Pembangun ES meningkatkan atau meningkatkan basis pengetahuan – Mitra
   - Seorang ahli - Kolega atau Asisten

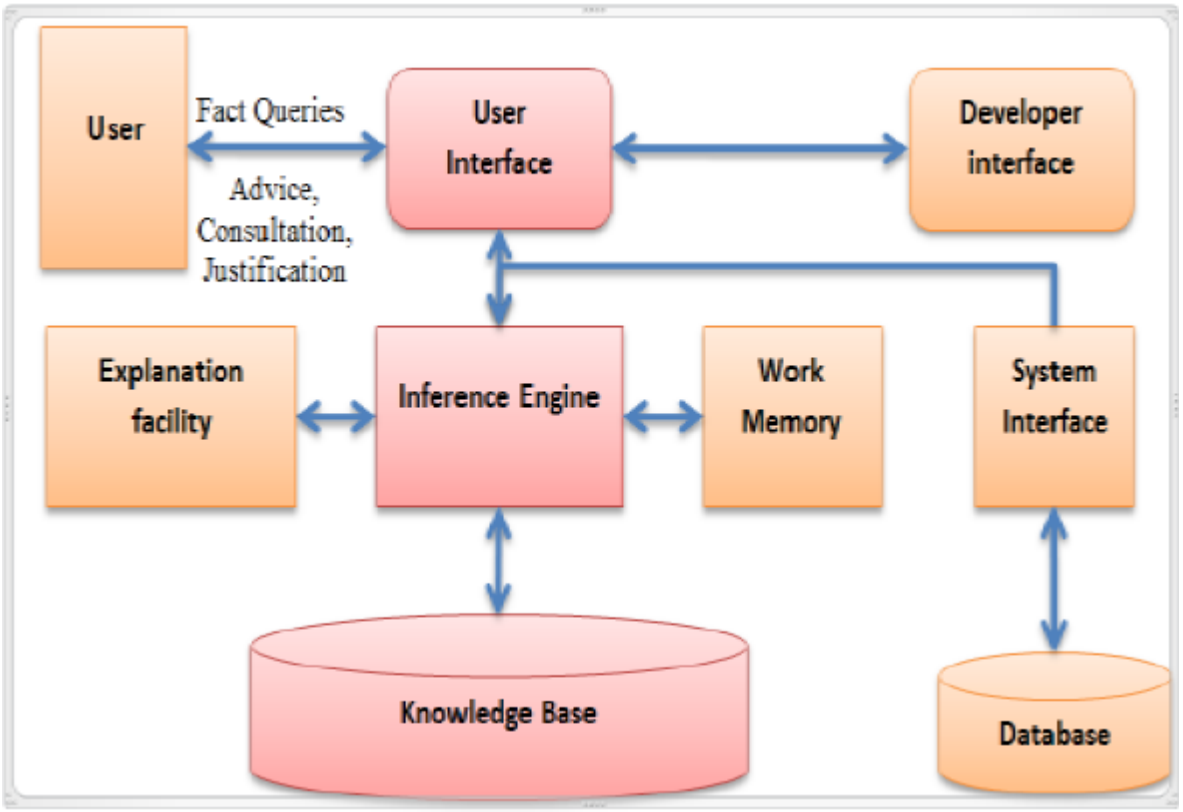
   Penting untuk dicatat bahwa pengembangan sistem pakar dapat dimulai ketika kelima pemain telah bergabung dengan tim. Namun, banyak sistem pakar sekarang dikembangkan pada komputer pribadi menggunakan cangkang sistem pakar. Ini dapat menghilangkan kebutuhan programmer dan juga dapat mengurangi peran insinyur pengetahuan. Untuk sistem pakar kecil, manajer proyek,

insinyur pengetahuan, programmer, dan bahkan pakar bisa menjadi orang yang sama. Tetapi semua pemain tim diperlukan ketika sistem pakar besar dikembangkan.

6.4 komponen sistem pakar, dan pengembangan sistem pakar

Sistem pakar biasanya dibangun untuk area aplikasi tertentu yang disebut Domain. Komponen Sistem Pakar adalah sebagai berikut:

- Antarmuka Pengguna.
- Mesin Inferensi.
- Basis pengetahuan.
- Memori kerja.
- Fasilitas penjelasan



Gambar 6.3 Arsitektur Sistem Pakar

1. Antarmuka Pengguna Adalah bagian dari sistem yang menerima kueri pengguna dalam bentuk yang dapat dibaca dan meneruskannya ke mesin inferensi kemudian menampilkan hasilnya kepada pengguna. Ini juga merupakan mekanisme dimana pengguna dan sistem berkomunikasi. Pemroses bahasa untuk komunikasi yang ramah dan berorientasi pada masalah Itu bisa memiliki menu dan grafik
2. Basis Pengetahuan: adalah kumpulan fakta dan aturan yang menggambarkan semua pengetahuan tentang domain masalah. Basis pengetahuan berisi pengetahuan yang diperlukan untuk memahami, merumuskan, dan memecahkan masalah Pengetahuan adalah bahan baku utama ES Dalam sistem pakar berbasis aturan, pengetahuan direpresentasikan sebagai seperangkat aturan. Setiap aturan menentukan relasi, rekomendasi, arahan, strategi atau heuristik dan memiliki struktur IF (kondisi) THEN (tindakan). Ketika bagian kondisi

dari suatu aturan terpenuhi, aturan tersebut dikatakan menyala dan bagian tindakan dieksekusi.

3. Mesin inferensi: melakukan penalaran dimana sistem pakar mencapai solusi. Ini menghubungkan aturan yang diberikan dalam basis pengetahuan dengan fakta-fakta yang disediakan dalam database.

- Membuat kesimpulan memutuskan aturan mana yang dipenuhi dan diprioritaskan.
- Otak ES
- Struktur kontrol (penerjemah aturan)
- Menyediakan metodologi untuk penalaran

Mesin inferensi mencoba memperoleh jawaban dari basis pengetahuan (memilih fakta dan aturan mana yang akan diterapkan ketika mencoba memecahkan kueri pengguna.
Ada dua jenis inferensi

a. Forward Chainning (Rantai ke depan)
b. Backfaerd Chainning (Rantai mundur)

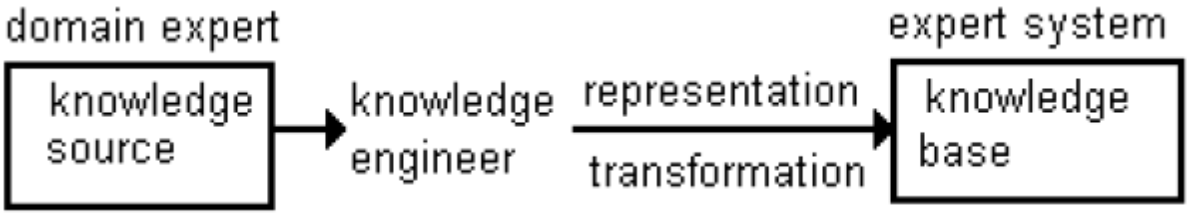
Forward Chainning (Rantai ke depan)
Mesin inferensi forward chaining mengambil aturan, dan jika kondisinya benar, menambahkan kesimpulannya ke memori kerja, sampai tidak ada lagi aturan yang bisa diterapkan; yaitu
jika kondisi aturan jika A dan B maka C,, benar,
kemudian
C ditambahkan ke memori kerja.
Dalam rangkaian ke depan, sistem hanya menguji aturan dalam urutan kemunculannya, oleh karena itu urutan aturan itu penting.
Rantai Mundur
Mesin inferensi rantai mundur mencoba membuktikan tujuan dengan menetapkan kebenaran kondisinya; yaitu
Aturan jika A dan B kemudian C,,, mesin rantai mundur akan mencoba membuktikan C dengan terlebih dahulu membuktikan A dan kemudian membuktikan B. Membuktikan kondisi ini benar, mungkin akan meminta panggilan lebih lanjut ke mesin dan seterusnya.

4. Memori kerja
   Memori kerja dapat digunakan untuk menyimpan kesimpulan antara dan informasi lain yang disimpulkan oleh sistem dari data.
5. Fasilitas penjelasan
   Fasilitas penjelasan memungkinkan pengguna untuk menanyakan sistem pakar bagaimana kesimpulan tertentu dicapai dan mengapa fakta tertentu diperlukan. Sistem pakar harus mampu menjelaskan alasannya dan membenarkan saran, analisis, atau kesimpulannya. Ini juga merupakan bagian dari sistem pakar yang memungkinkan pengguna atau pengambil keputusan untuk memahami bagaimana sistem pakar sampai pada kesimpulan atau hasil tertentu.

6.5 Pengembangan Sistem Pakar

Proses membangun sistem pakar disebut rekayasa pengetahuan. Sejalan dengan itu, pengembang sistem pakar disebut sebagai insinyur pengetahuan.
Rekayasa pengetahuan atau rekayasa berbasis pengetahuan
Teknologi sistem pakar adalah varietas pendekatan kecerdasan buatan (AI) di mana pengetahuan pengambilan keputusan dikodifikasi dan dimodelkan, proses merancang sistem pakar melalui pendekatan AI disebut rekayasa berbasis pengetahuan.

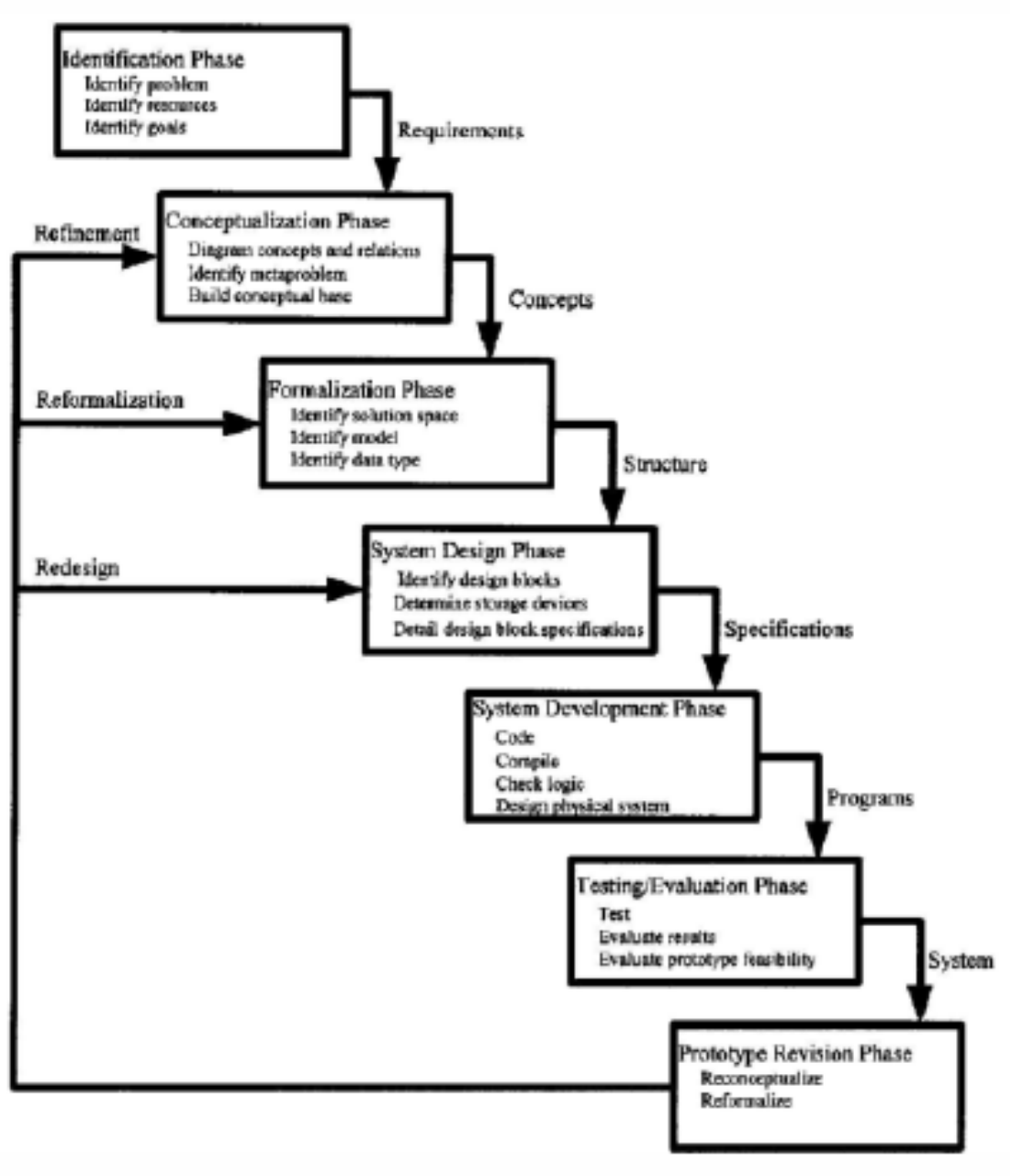
Proses membangun sebuah sistem pakar adalah sebuah usaha untuk menangkap keahlian yang langka atau penting dan mewujudkannya dalam kode komputer. Proses ini melibatkan ketelitian kloning intelektual untuk menyuntikkan pengetahuan ahli ke dalam objek buatan. Pembangun sistem pakar, insinyur pengetahuan, mencari tahu dari para ahli apa yang mereka ketahui dan bagaimana mereka menggunakan pengetahuan mereka untuk memecahkan masalah tertentu. Setelah pembekalan ini dilakukan, pembuat sistem pakar menggabungkan pengetahuan dan keahlian dalam program komputer, membuat pengetahuan dan keahlian mudah direplikasi, mudah didistribusikan, dan pada dasarnya abadi.
Ini terdiri dari tiga tahap:
1. Perolehan pengetahuan: proses memperoleh pengetahuan dari para ahli (dengan mewawancarai dan/atau mengamati pakar manusia, membaca buku-buku tertentu, dll).
2. Representasi pengetahuan: memilih struktur yang paling tepat untuk mewakili pengetahuan (daftar, set, skrip, pohon keputusan, triplet objek-atribut-nilai, dll).
3. Validasi pengetahuan: menguji bahwa pengetahuan ES benar dan lengkap



Gambar 6.4: Struktur proses rekayasa pengetahuan

Konsep rekayasa berbasis pengetahuan tumbuh dari pekerjaan awal pada sistem pakar di tahun tujuh puluhan. Dengan semakin populernya sistem berbasis pengetahuan, muncul juga kebutuhan untuk pendekatan sistematis untuk membangun sistem seperti itu, mirip dengan metodologi dalam rekayasa perangkat lunak arus utama. Selama bertahun-tahun, disiplin rekayasa pengetahuan telah berkembang menjadi pengembangan teori, metode, dan alat untuk mengembangkan aplikasi intensif pengetahuan. Dengan kata lain, ini memberikan panduan tentang kapan dan bagaimana menerapkan teknik presentasi pengetahuan tertentu untuk memecahkan masalah tertentu.
Dalam kursus ini, dua aspek terkait rekayasa pengetahuan akan dijelaskan; ini adalah pengembangan berulang dan pembuatan prototipe.
Sistem pakar dikembangkan secara iteratif dalam serangkaian langkah yang berulang. Langkah-langkah ini secara kasar terdiri dari akuisisi pengetahuan, diikuti oleh desain sistem (atau modifikasi desain yang ada, pengembangan sistem (termasuk entri pengetahuan) dan pengujian dan evaluasi sistem.

Langkah-langkah dalam analisis sistem pakar dan metodologi desain diringkas di bawah ini.



Gambar 6.5: Pengembangan Iteratif Sistem Pakar

4. Fase Identifikasi

Langkah pertama dalam fase identifikasi, Identifikasi masalah, mirip dengan fase definisi masalah dalam siklus hidup pengembangan sistem tradisional. Tujuannya adalah untuk mengidentifikasi, mengkarakterisasi, dan mendefinisikan masalah yang diharapkan dapat dipecahkan oleh sistem dan kemudian membagi masalah tersebut menjadi sub-tugas yang sesuai. Setelah masalah didefinisikan, sumber daya yang diperlukan untuk memperoleh pengetahuan, mengimplementasikan sistem, dan menguji sistem diidentifikasi. Sumber daya yang umum termasuk pengetahuan, waktu, fasilitas komputasi, dan uang. Karena sistem pakar mahal dan membuatnya membutuhkan waktu yang cukup lama, studi kelayakan sering dilakukan sebelum pekerjaan berkembang melampaui titik ini. Selain mengidentifikasi sumber daya, analis dan/atau perancang sistem pakar juga mengidentifikasi tujuan dan sasaran sistem. Sangat membantu untuk

mengidentifikasi dan mendokumentasikan tujuan secara eksplisit karena pendekatan desain tertentu, seperti pencarian heuristik, pencarian luas, pencarian mendalam, dan penalaran didorong oleh tujuan.

2. Fase konseptualisasi

   Tugas utama dari fase konseptualisasi adalah untuk membuat diagram dari konsep-konsep kunci sistem dan hubungan untuk mendefinisikan basis konseptual untuk sistem prototipe. Tujuan utama termasuk memisahkan mesin inferensi dari domain masalah, memfaktorkan (menganalisis) masalah ke dalam meta-masalah, mengidentifikasi konsep dan hubungan kunci sistem, dan menguji konsep dan hubungan tersebut dengan menantang mereka (dengan contoh spesifik pemecahan masalah kegiatan) untuk memastikan bahwa mereka mencakup setiap kasus umum. Banyak alat dan teknik yang digunakan dalam fase ini.

3. Fase Formalisasi

   Fase formalisasi melibatkan pemetaan konsep-konsep kunci, sub-masalah, dan karakteristik aliran informasi yang diisolasi selama konseptualisasi menjadi representasi yang lebih formal berdasarkan berbagai rekayasa pengetahuan dan alat pemecahan masalah dan kerangka kerja representasi pengetahuan. Tujuan utamanya adalah untuk mengidentifikasi solusi ruang (domain dengan kumpulan semua solusi yang mungkin), ruang hipotesis (ruang solusi hipotetis), model yang mendasari, dan karakteristik data. Untuk menentukan struktur ruang hipotesis, analis atau perancang sistem harus memformalkan konsep (pengetahuan dalam format abstrak yang dapat digunakan untuk memandu proses pencarian atau penalaran) dan menentukan bagaimana mereka digabungkan untuk membentuk hipotesis. Konsep memberikan petunjuk tentang sifat ruang seperti jika terbatas, jika hierarki harus dipertimbangkan, jika tingkat abstraksi tertentu dapat diterapkan, dan jika kelas tertentu dari konsep harus dihasilkan. Teknik pencarian seperti pencarian buta, pencarian heuristik, dan abstraksi ruang solusi sering digunakan. Teknik penalaran seperti membangun asumsi, membangun pembenaran, dan kendala dan teknik tujuan membantu mengidentifikasi model yang mendasari proses yang digunakan untuk menghasilkan solusi dalam domain.

4. Fase desain sistem

   Selama fase desain sistem (kadang-kadang disebut fase desain logis), analis dan/atau perancang menentukan bagaimana sistem akan memenuhi persyaratan yang diidentifikasi selama tiga fase sebelumnya. Biasanya, laporan dan keluaran lain yang harus dihasilkan sistem ditentukan terlebih dahulu. Fase ini mirip dengan tahap desain dalam siklus hidup pengembangan sistem tradisional. Perhatikan, bagaimanapun, bahwa skema representasi yang digunakan untuk menggambarkan pengetahuan berbeda dari metodologi tradisional. Dengan menggunakan pengetahuan yang telah Anda peroleh dan alat yang telah Anda pilih, Anda sekarang dapat memulai desain sistem pakar. Pertama, Anda perlu membuat garis besar, bagan alur hierarki, matriks, tabel keputusan, atau format lain yang akan membantu Anda mengatur dan memahami pengetahuan. Dengan menggunakan bantuan ini, Anda akan mengubah pengetahuan menjadi aturan IFTHEN. Yang terbaik adalah mengikuti prosedur khusus yang direkomendasikan oleh perangkat lunak Anda menggunakan. Setelah desain dasar selesai, Anda dapat mulai menggunakan alat untuk membuat prototipe dari satu segmen sistem. Terjemahkan sebagian pengetahuan menjadi aturan dan uji segmen yang baru dibuat. Uji konsep sebelum melanjutkan seluruh program.

5. Fase pengembangan sistem

   Prototipe sistem pakar dibuat selama tahap pengembangan sistem (atau desain fisik). Tahap ini mirip dengan tahap pengembangan dalam siklus hidup pengembangan sistem tradisional. Setelah Anda meyakinkan diri sendiri bahwa sistem akan bekerja dengan memuaskan, sekarang Anda dapat mulai mengembangkan prototipe ke dalam sistem final. Cara terbaik untuk melakukannya adalah dengan mengembangkan prototipe satu segmen pada satu waktu.

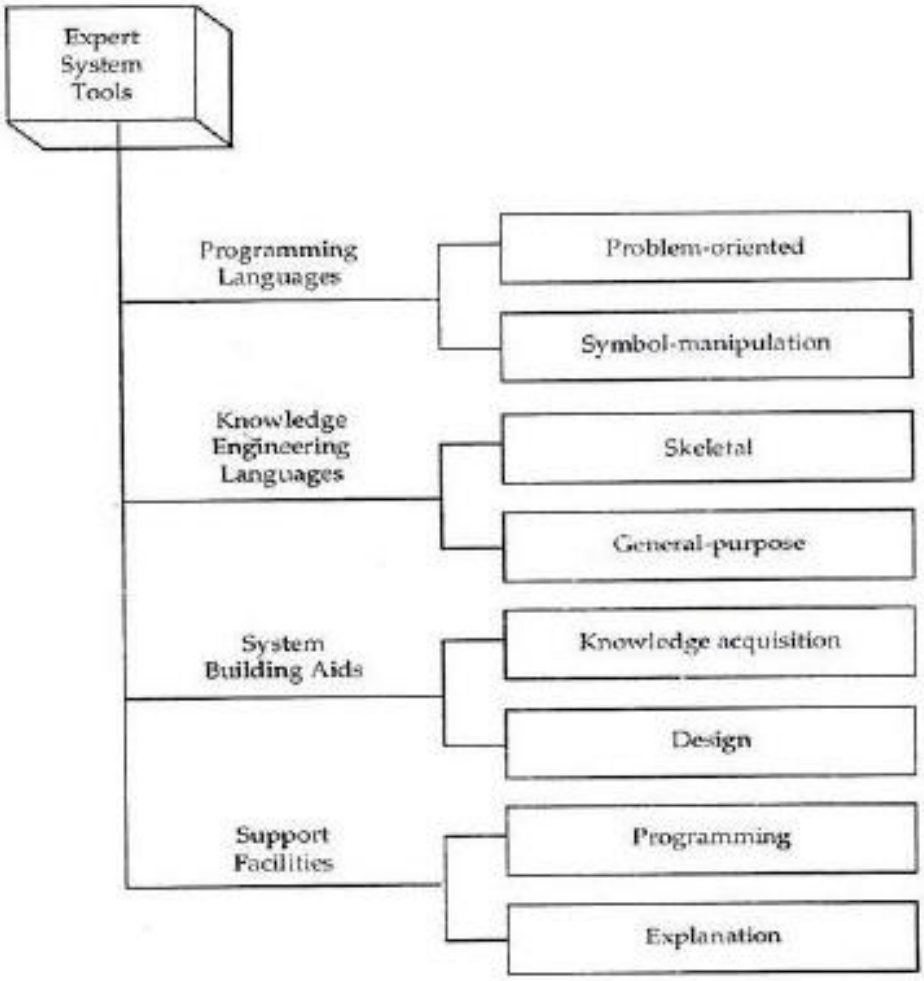
6. Tahap pengujian dan evaluasi
   Selama fase ini, sistem prototipe dievaluasi. Fase ini sejajar dengan tahap pengujian dalam siklus hidup pengembangan sistem tradisional. Namun, selain alat dan teknik pengujian, sistem pakar menggunakan teknik pengujian dinamis untuk memverifikasi proses penalaran dan/atau inferensi. Setelah sistem pakar dikembangkan, Anda perlu meluangkan waktu untuk menguji dan men-debug-nya. Tidak ada sistem pakar yang sempurna untuk pertama kalinya, dan banyak pekerjaan akan diperlukan untuk memvalidasinya. Umpan balik pengguna akan menunjukkan kepada Anda di mana harus membuat perubahan akhir, koreksi, dan penambahan untuk mencapai kinerja yang diinginkan.
7. Fase revisi prototipe
   Sebuah sistem pakar berkembang dari waktu ke waktu, menyerukan revisi hampir konstan, sistem pakar sifat berbagi dengan sebagian besar prototipe. Berdasarkan hasil tahap pengujian/evaluasi, konsep dan relasi disempurnakan, ruang solusi, model, karakteristik data diformalkan kembali, dan sistem didesain ulang.
8. Menjaga Sistem
   Bagian penting dari pengembangan sistem pakar adalah pemeliharaan berkelanjutan, memperbarui sistem dengan pengetahuan baru, menghapus pengetahuan yang tidak lagi berlaku, dan menyempurnakan sistem agar tetap terkini dan dapat diterapkan pada masalah.

6.5.1 Pengembangan sistem pakar Alat perangkat lunak

Sebagian besar sistem pakar dikembangkan melalui perangkat lunak khusus yang disebut shell. Shell menyediakan kerangka kerja untuk menghasilkan sistem pakar. Jadi basis pengetahuan dan aturan hanya ditambahkan ke kerangka kerja ini. Shell ini dilengkapi dengan mekanisme inferensi (backward chaining, forward chaining, atau keduanya), dan membutuhkan pengetahuan untuk dimasukkan sesuai dengan format yang ditentukan. Mereka biasanya datang dengan sejumlah fitur lain, seperti alat untuk menulis hypertext, untuk membangun antarmuka pengguna yang ramah, untuk memanipulasi daftar, string, dan objek, dan untuk antarmuka dengan program dan database eksternal. Shell ini memenuhi syarat sebagai bahasa, meskipun tentu saja dengan jangkauan aplikasi yang lebih sempit daripada kebanyakan bahasa pemrograman.

Memilih alat yang tepat untuk membangun sistem pakar merupakan langkah penting. Terkadang, lebih dari satu alat akan digunakan selama proyek berlangsung. Biasanya, satu alat digunakan untuk pembuatan prototipe, tetapi alat yang berbeda dipilih untuk pengembangan dan pengiriman skala besar.

Pemrograman sistem pakar dan alat menyederhanakan pekerjaan membangun sistem pakar. Mulai dari bahasa pemrograman tingkat tinggi hingga fasilitas pendukung tingkat rendah. Ini dibagi menjadi empat kategori utama



Gambar 6.6: Menunjukkan jenis alat untuk desain sistem pakar

6.6 Kebutuhan Sistem Pakar dan Aplikasi

Di banyak organisasi, keahlian pemecahan masalah sangat langka. Melatih orang untuk menjadi mahir dalam memecahkan masalah khusus membutuhkan waktu dan membutuhkan investasi yang besar. Oleh karena itu, para ahli selalu kekurangan pasokan. Dalam pengaturan operasional, frekuensi terjadinya masalah sering kali melebihi kemampuan sejumlah pakar yang terbatas. Dalam beberapa situasi, ahli mungkin secara geografis jauh dari lokasi masalah, atau masalah dapat terjadi selama ahli tidak tersedia. Akibatnya, masalah harus ditangani oleh personel yang kurang berkualitas. Hal ini dapat menyebabkan penundaan dan menyebabkan pengambilan keputusan yang tidak konsisten atau tidak merata. Salah satu contohnya adalah lantai pabrik di mana masalah peralatan yang sangat khusus harus didiagnosis dan perbaikan dilakukan. Biasanya, tugas ini dilakukan oleh seorang ahli terlatih yang memiliki pengalaman bertahun-tahun. Pekerjaan berjalan normal jika frekuensi malfungsi relatif rendah dan ahli siap tersedia. Namun, sebuah perusahaan mungkin memiliki pabrik di lokasi yang berbeda, atau pabrik dapat beroperasi sepanjang waktu. Dalam keadaan ini, permintaan akan ahli dapat dengan cepat melebihi ketersediaannya, yang mengakibatkan penundaan dan masalah. Salah satu solusinya adalah mengembangkan sistem pakar untuk membantu mengidentifikasi kerusakan mesin yang sering terjadi dan menyarankan solusi. Sistem pakar semacam itu dapat digunakan di lantai pabrik dan digunakan untuk memecahkan masalah rutin yang seharusnya ditangani oleh pakar. Jika sistem pakar dapat memecahkan masalah yang biasanya dipecahkan

oleh pakar, penundaan dapat dihilangkan dan produktivitas dapat ditingkatkan. Salinan sistem pakar dapat didistribusikan ke seluruh perusahaan, membuat keahlian tersedia di beberapa lokasi sepanjang waktu.
Ini adalah deskripsi sederhana dari penggunaan produktif sistem pakar. Tentu saja, ada potensi masalah dalam skenario ini yang dapat mengalahkan penggunaan teknologi sistem pakar secara efektif
Berikut adalah faktor-faktor yang menyarankan sistem pakar sesuai.

- Kebutuhan membenarkan biaya dan usaha
- Keahlian manusia tidak selalu tersedia
- Masalah membutuhkan penalaran simbolis
- Domain masalah terstruktur dengan baik
- Metode komputasi tradisional gagal
- Ada ahli yang kooperatif dan pandai berbicara
- Masalah tidak terlalu besar

6.6.1 Area Aplikasi Sistem Pakar

Sistem pakar telah memperoleh aplikasi yang lebih luas di berbagai bidang usaha manusia, terutama dalam peran di mana keahlian manusia dibutuhkan. Beberapa area di mana sistem pakar diterapkan dibahas di bawah ini:

1. Akuntansi & Keuangan
   Industri jasa keuangan telah menjadi pengguna yang kuat dari teknik sistem pakar. Program penasihat telah dibuat untuk membantu para bankir dalam menentukan apakah akan memberikan pinjaman kepada bisnis dan individu. Perusahaan asuransi telah menggunakan sistem pakar untuk menilai risiko yang disajikan oleh pelanggan dan untuk menentukan harga asuransi. Aplikasi khas di pasar keuangan adalah dalam perdagangan valuta asing. Area lain dalam aplikasi akuntansi dan keuangan adalah - Pemilih Kode Biaya, Perdagangan Saham & Komoditas, Konstruksi Portofolio, Pembelian Rumah, Pelatihan dan Seleksi Perencana Keuangan, Penasihat Pajak Pribadi, Mendeteksi Perdagangan Orang Dalam, Layanan Organisasi, Penasihat Analisis Kredit, Identifikasi Pinjaman Bank, Kredit Pengendalian, Dokumentasi Pinjaman, Penilaian Risiko dan Penipuan di Lembaga Keuangan, Bantuan Pengisian Formulir Pajak, Prediktor Persetujuan Pinjaman Komersial...
2. Pertanian - Irigasi dan Pengendalian Hama, Pemilihan dan Pengelolaan Varietas Tanaman, Karakterisasi dan Pemanfaatan Tanah untuk Area Tertentu, Pemupukan, Iklim dan Interaksi dan Analisis Tanah, Tingkat Stok Ikan Salmon dan Seleksi Spesies, Inventarisasi Hutan, Identifikasi Gulma, Konservasi Tanah, Pemilihan Pohon Berdasarkan Kondisi Lingkungan, Perencanaan dan Desain Sistem Agroforestry.
3. Bisnis - Alternatif untuk Industri Terfragmentasi, Pengembangan Salinan Periklanan, Dokumentasi dan Rute Pengiriman, Penasihat Pasar untuk Sistem Kontrol Proses, Penilaian Demografis dan Pasar, Pemecahan Masalah Kinerja Produk, Penilaian Tenaga Penjualan, Pemasaran Akun, Kemampuan Paten Penemuan, Gaji & Manfaat Perencanaan, Pemilihan Aplikasi Bisnis Profil Klien, Pemilihan Layanan Profesional, Perencanaan Sasaran Karir, Kalkulator Dana Pensiun, Kelayakan Asuransi Pengangguran.
4. Evaluasi Bahaya Bahan Kimia, Prosedur Fasilitas Kimia, Campuran Bahan Propelan yang Benar, Izin Teknologi Pengendalian Polusi, Identifikasi Logam dan Paduan Biasa, Proses Real-time Terkendali Pengelolaan Air Limbah Kota, Pemilihan Pelarut untuk Senyawa Kimia, Resep dan Identifikasi Glasir Tembikar, Pulp Penasihat Pemutihan, Toksisitas Bahan Kimia Laboratorium, Sistem Pengenalan Kapur, Diagnosis Proses dan Pemecahan Masalah...
5. Pemodelan Diagnostik Sistem Komputer - Perangkat Lunak, Ukuran Aplikasi, Jaminan Kualitas Perangkat Lunak, Klasifikasi Program, Lokasi Kegagalan & Analisis Komponen,

Pemilihan Teknologi Baru, Sistem Pelatihan, Diagnostik Perangkat Keras Kustom, Sistem Pendukung Keputusan, Sistem Pendukung MIS, Pemeliharaan Pustaka Program, Deteksi Kesalahan dan Diagnostik Jaringan Area Luas, Analisis Data Statistik, Konfigurasi Personal Computer, Pemilihan Teknik Perisai, Desain Basis Data, Representasi Layanan Teknis Terkomputerisasi, Pemilihan Perangkat Keras dan Perangkat Lunak oleh Pengguna Non-Teknis, Pengguna Baru Bantuan Komputer, Rekomendasi Dokumentasi kepada Pengguna, Pemantauan , Bantuan Perbaikan dan Prediksi Masalah Sistem Operasi.

6. Konstruksi - Rehabilitasi & Desain Perkerasan, Penilaian Kerusakan Struktural, Evaluasi & Pemilihan Peralatan, Penetapan Biaya & Pemilihan Material, Penjadwalan Proyek, Perkiraan Biaya, Evaluasi Proyek Perumahan Multikeluarga, Pelatih/Penasihat Keselamatan Zona Kerja, Pemilihan Prosedur Pengelasan dan Perkiraan Biaya, Pemadatan Tanah , Penasihat Kode Kebakaran, Sistem Manajemen Alarm.
7. Pendidikan - Rekomendasi Bahan Referensi Perpustakaan, Interpretasi Data Kontrol Kualitas Statistik, Pengajaran Mineral, Identifikasi Batuan dan Fosil, Kelayakan Bantuan Keuangan Mahasiswa, Analisis Awas Logam, Penasihat Manajemen Darurat Departemen Pemadam Kebakaran, Sistem Diagnostik Mahasiswa Kedokteran, Penasihat Kedokteran Gigi, Pelanggan Telepon Instruksi Pendukung, Pelatihan Turbin Gas, Pelatihan Industri, Penasihat Perawatan Pasien untuk Perawat Mahasiswa.
8. Rekayasa, Sistem pakar banyak digunakan dalam rekayasa, mereka dapat membantu dalam konfigurasi, dimana solusi untuk suatu masalah disintesis dari satu set elemen yang terkait dengan satu set kendala. Teknik ini telah menemukan jalannya untuk digunakan di banyak industri teknik yang berbeda, misalnya, pembangunan rumah modular, manufaktur, dan masalah lain yang melibatkan desain dan manufaktur teknik yang kompleks. Area lain dari aplikasi ES di bidang teknik adalah:
   Kontrol Perubahan Teknik, Analisis Keausan Oli Pelumas Mesin Diesel, Analisis Fase Paduan Super, Diagnostik Peralatan, Pengujian Kegagalan Semikonduktor Elektronik, Pemilihan & Ukuran Daftar Suku Cadang, Penjadwalan Sistem Pembangkit Listrik, Prediksi Kegagalan Komponen, Penasihat Pemesinan, Pemilihan Alat Mesin yang Dikontrol Secara Numerik , Kontrol Proses Pabrik Petrokimia, Desain Tata Letak Panel Kontrol, Penasihat Desain Material dan Proses...
9. Asuransi - Peringkat untuk Asuransi Jiwa di Bawah Standar, Klasifikasi Komp Pekerja, Bantuan Penjaminan, Help Desk dan Identifikasi Manfaat Jaminan Sosial, Kelayakan Asuransi Pengangguran.
10. Medis: Sistem pakar memiliki kehadiran yang kuat dalam aplikasi medis. Perlu dicatat bahwa Diagnosis medis adalah salah satu bidang pengetahuan pertama di mana teknologi ES diterapkan. Aplikasi medis lain dari sistem pakar adalah: Protokol Penerimaan, Analisis X-Ray, Diagnosis Hematologi, Wawancara Psikiatri, Interpretasi Respons Batang Otak Auditori Pediatrik, Pengambilan Keputusan Medis, Pemilihan Respirator untuk Anak Prasekolah, Pemanfaatan dan Pemodelan Layanan Kesehatan, Sistem Diagnostik, In-Vitro Fertilisasi, Analisis Gejala, Diagnosis Lab Berbasis Suara, Strategi Kelayakan Rehabilitasi, Manajemen Penagihan dan Akun, Penelitian Penyakit. Dll
11. Terkait Militer, Pemerintah & Luar Angkasa - Pelatihan Petugas Pendekatan Kapal Selam, Pemilihan Metodologi Tempur, Perancangan Stasiun Kerja Mode Radar, Manajemen Kontrak Federal, Prakiraan Cuaca Buruk, Analisis On-Orbit Payload Antar-Jemput, Pemilih Material Logam, Koreksi Abnormalitas Satelit GB, Analis Termal, Pemilihan Non-Material dalam Aplikasi Dirgantara...

12. Pemecahan Masalah
    Kelas ini terdiri dari sistem yang menyimpulkan kesalahan dan menyarankan tindakan korektif untuk perangkat atau proses yang tidak berfungsi. Ini memiliki aplikasi luas di berbagai bidang seperti:
    - Sistem Start Pesawat, Komunikasi Data, Pengujian dan Perbaikan PCB, Sistem Kontrol Turbin Gas, Kegagalan Sistem Bearing, Kesulitan Telekomunikasi, Kontrol Proses Waktu Nyata, Sistem Asisten Meterman, Diagnosis Peralatan dan Sistem Mekanik, Deteksi Cacat Las, Diagnosis Kerusakan Web dalam Penggilingan Kertas, Sistem Pemeliharaan Bearing Generator Turbin Pembangkit Listrik.
13. Penerbitan Pengetahuan: Ini adalah area yang relatif baru, tetapi juga berpotensi meledak. Fungsi utama dari sistem pakar adalah untuk menyampaikan pengetahuan yang relevan dengan masalah pengguna, dalam konteks masalah pengguna. Dua sistem pakar yang paling banyak didistribusikan di dunia termasuk dalam kategori ini. Yang pertama adalah penasihat yang menasihati pengguna tentang penggunaan tata bahasa yang tepat dalam sebuah teks. Yang kedua adalah penasihat pajak yang menyertai program persiapan pajak dan memberi saran kepada pengguna tentang strategi, taktik, dan kebijakan pajak individu.

Banyak, Banyak Lainnya - Pengonfigurasi Sistem Telepon, Materi Pelatihan & Pemilihan Produk, Perencanaan Sumber Daya Manufaktur, Penjadwalan Produksi, Jaringan Layanan, Penjadwalan Maskapai Penerbangan, Analisis Biaya/Manfaat, Implementasi Perencanaan, Pengembangan Karir, Estimasi Proposal Cepat, Kontrol Kredit, Pengembangan Produk, Telepon Penyaringan Panggilan, Analisis Pasar Real Estat & Kredit Hipotek, Perencanaan Pensiun, Analisis Penjualan.

Spektrum aplikasi teknologi sistem pakar untuk masalah industri dan komersial sangat luas sehingga tidak mudah untuk dikarakterisasi. Aplikasi menemukan jalan mereka ke sebagian besar bidang pekerjaan pengetahuan. Mereka beragam seperti membantu tenaga penjualan menjual rumah modular yang dibangun pabrik hingga membantu NASA merencanakan pemeliharaan pesawat ulang-alik sebagai persiapan untuk penerbangan berikutnya.

6.7 Representasi Pengetahuan dalam sistem pakar

Representasi Pengetahuan adalah bidang dalam kecerdasan buatan yang berfokus pada menangkap informasi tentang dunia yang dapat digunakan untuk memecahkan masalah kompleks dalam domain tertentu seperti mendiagnosis kondisi medis.

Pengetahuan dan Representasi adalah dua entitas yang berbeda. Mereka memainkan peran sentral tetapi dapat dibedakan dalam desain sistem cerdas.

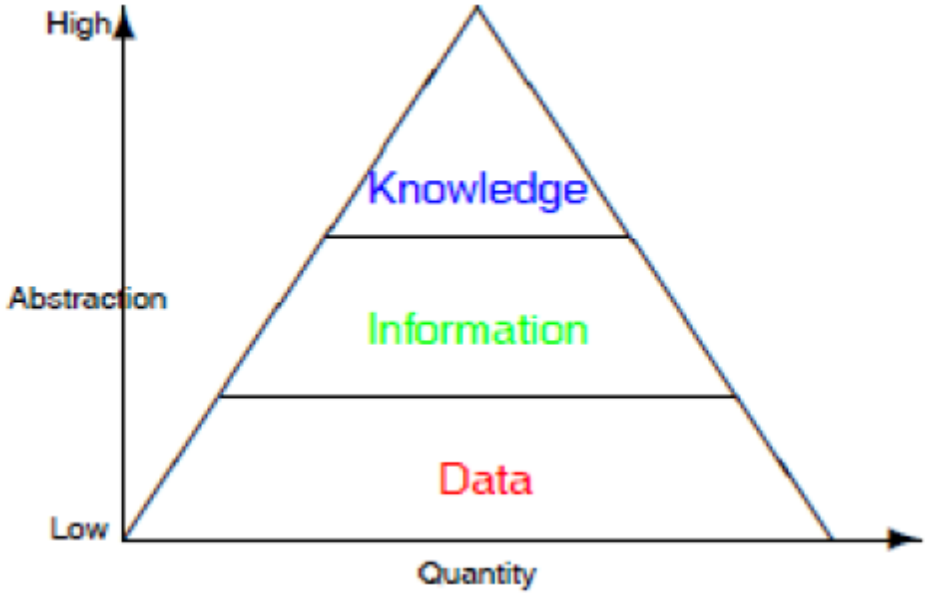
Pengetahuan adalah deskripsi dunia. Ini menentukan kompetensi sistem dengan apa yang diketahuinya. Di sisi lain Representasi adalah cara pengetahuan dikodekan sehingga komputer dapat memahami bagaimana memecahkan suatu masalah. Ini mendefinisikan kinerja sistem dalam melakukan sesuatu.

Pengetahuan adalah perkembangan yang dimulai dengan data yang kegunaannya terbatas.

1. Data dipandang sebagai kumpulan fakta yang tidak terhubung
   Contoh: Hujan turun
2. Dengan mengatur atau menganalisis data, kita memahami apa itu data
   berarti, dan ini menjadi informasi. Ini memberikan jawaban untuk "siapa", "apa", "di mana", dan "kapan".
   Contoh: Suhu turun 15 derajat dan kemudian hujan turun
3. Interpretasi atau evaluasi informasi menghasilkan pengetahuan. Ini memberikan jawaban sebagai "bagaimana".
   Contoh: Jika kelembaban sangat tinggi dan suhu turun secara substansial, maka atmosfer tidak mungkin menahan kelembaban, sehingga hujan

4. Pemahaman tentang prinsip-prinsip yang terkandung dalam pengetahuan adalah kebijaksanaan. Ini memberikan jawaban sebagai "mengapa".

Pengetahuan adalah yang paling abstrak dan ada dalam jumlah terkecil. Pengetahuan itu sendiri dapat memiliki tingkatan abstraksi: konkret (pengetahuan tentang masalah tertentu), domain khusus (kelas masalah) dan abstrak (banyak kelas masalah).



Gambar 6.7: Menunjukkan tingkat abstraksi pengetahuan dari data

6.7.1 Mengapa kita membutuhkan Representasi pengetahuan?

Representasi pengetahuan sangat penting karena pemecahan masalah membutuhkan sejumlah besar pengetahuan dan beberapa di antaranya harus menjadi mekanisme untuk memanipulasi pengetahuan itu ke dalam bentuk yang dapat dimengerti komputer.

Bagaimana kita merepresentasikan apa yang kita ketahui?

Mewakili pengetahuan memerlukan analisis untuk membedakan antara pengetahuan “bagaimana” dan pengetahuan “itu.

Mengetahui "bagaimana melakukan sesuatu".

misalnya "cara mengemudi mobil" adalah pengetahuan prosedural

Mengetahui "bahwa sesuatu itu benar atau salah".

misalnya "Itu adalah batas kecepatan untuk mobil di jalan raya" adalah pengetahuan Deklaratif.

Pengetahuan dikategorikan ke dalam dua jenis utama: Tacit dan Explicit Istilah “Tacit” sesuai dengan jenis pengetahuan "informal" atau "implisit" yang tidak dikodifikasi.

1. Ada dalam diri manusia; itu diwujudkan.
2. Sulit untuk diartikulasikan secara formal
3. Sulit untuk berkomunikasi atau berbagi
4. Sulit untuk dicuri atau disalin.
5. Diambil dari pengalaman, tindakan, wawasan subjektif.

Di sisi lain istilah "Eksplisit" sesuai dengan jenis pengetahuan "formal" yang dikodifikasi.

1. Ada di luar manusia; itu tertanam.
2. Dapat diartikulasikan secara formal
3. Dapat dibagikan, disalin, diproses, dan disimpan
4. Mudah dicuri atau disalin

5. Diambil dari artefak dari beberapa jenis sebagai prinsip, prosedur, proses, konsep
Peta Tipologi Pengetahuan
Ini menunjukkan hubungan antara – Pengetahuan Tacit dan Eksplisit.
Pengetahuan tacit berasal dari “pengalaman”, “tindakan”, “subyektif”, “wawasan”.
Pengetahuan eksplisit berasal dari “prinsip”, prosedur, “proses”, “konsep”.
Fakta: adalah data atau kejadian yang spesifik dan unik.
Konsep: adalah kelas item, kata, atau ide yang dikenal dengan nama umum dan berbagi fitur umum.
Proses: adalah aliran peristiwa atau aktivitas yang menggambarkan bagaimana sesuatu bekerja daripada bagaimana melakukan sesuatu. Prosedur: adalah serangkaian tindakan dan keputusan langkah demi langkah yang menghasilkan pencapaian tugas.
Prinsip: adalah pedoman, aturan, dan parameter yang mengatur; prinsip memungkinkan untuk membuat prediksi dan menarik implikasi;
Representasi pengetahuan yang baik memungkinkan akses yang cepat dan akurat ke pengetahuan dan pemahaman konten.

6.7.2 Metode Representasi Pengetahuan

Pengetahuan diwakili oleh tiga metode dalam sistem pakar; ini adalah:
I. Aturan Produksi
II. jaringan semantik
III. Frame dan Logika
Aturan Produksi atau Sistem Produksi:
Aturan digunakan untuk mewakili hubungan. Representasi pengetahuan berbasis aturan mempekerjakan
Kondisi IF (premis atau konsekuen)
THEN tindakan (tujuan atau anteseden) pernyataan.
Sebagai contoh,
JIKA elemen pemanas menyala DAN roti selalu gelap
MAKA termostat pemanggang roti rusak
Ketika situasi masalah cocok dengan bagian IF dari aturan, tindakan yang ditentukan oleh bagian THEN dari aturan dilakukan
- Aturan produksi adalah salah satu bahasa representasi pengetahuan yang paling populer dan banyak digunakan
- Sistem aturan produksi terdiri dari tiga komponen
I. Memori kerja berisi informasi yang diperoleh sistem tentang masalah sejauh ini.
II. Basis aturan berisi informasi yang berlaku untuk semua masalah yang mungkin diminta untuk dipecahkan oleh sistem.
III. Interpreter memecahkan masalah kontrol, yaitu, memutuskan aturan mana yang akan dieksekusi pada setiap pilihan - eksekusi siklus.

Keuntungan dari metode aturan Sistem Produksi:
Kealamian ekspresi
Modularitas
Sintaks terbatas
Kemampuan untuk Mewakili Pengetahuan yang Tidak Pasti
Kekurangan Sistem Produksi:
Tidak efisien
Kurang ekspresif

jaringan semantik
Ini adalah formalisme/mekanisme untuk merepresentasikan informasi/Pengetahuan tentang
objek, orang, konsep, dan hubungan khusus di antara mereka.
Sintaks jaring semantik sederhana. Ini adalah jaringan node dan link berlabel.

Ini adalah grafik berarah dengan simpul yang sesuai dengan konsep, fakta, objek, dll.
dan busur yang menunjukkan hubungan atau asosiasi antara dua konsep.

Tautan yang umum digunakan dalam jaring semantik adalah dari jenis berikut.
isa → subkelas entitas (mis., rumah sakit anak adalah subkelas rumah sakit)
inst → instance tertentu dari suatu kelas (mis., India adalah turunan dari negara)
prop → tautan properti (mis., properti anjing adalah bark)
Representasi Pengetahuan dalam Semantic Net
Setiap manusia, hewan dan burung adalah makhluk hidup yang bernafas dan makan. Semua burung bisa terbang.
Semua pria dan wanita adalah manusia yang memiliki dua kaki. Kucing adalah hewan dan memiliki bulu.
Semua hewan memiliki kulit dan dapat bergerak. Jerapah adalah hewan yang bertubuh tinggi dan memiliki kaki yang panjang.
Parrot adalah burung dan berwarna hijau
Warisan dalam Semantic Net
-Mekanisme pewarisan memungkinkan pengetahuan disimpan pada tingkat abstraksi setinggi mungkin yang mengurangi ukuran basis pengetahuan.
- Ini memfasilitasi inferensi informasi yang terkait dengan jaring semantik.
- Ini adalah alat alami untuk mewakili informasi yang terstruktur secara taksonomi dan memastikan bahwa semua anggota dan sub-konsep dari suatu konsep memiliki sifat yang sama.
- Ini juga membantu kita menjaga konsistensi basis pengetahuan dengan menambahkan konsep baru dan anggota dari yang sudah ada.
- Properti yang melekat pada objek (kelas) tertentu harus diwarisi oleh semua subkelas dan anggota kelas itu.

Keuntungan dari jaring semantik
 Mudah divisualisasikan
 Definisi formal dari jaringan semantik telah dikembangkan
 Pengetahuan terkait mudah dikelompokkan.
 Efisien dalam kebutuhan ruang
 Objek direpresentasikan hanya sekali
 Hubungan ditangani oleh pointer
Kekurangan jaring semantik
 Warisan (terutama dari berbagai sumber dan ketika pengecualian dalam pewarisan diinginkan) dapat menyebabkan masalah.
 Fakta yang ditempatkan secara tidak tepat menyebabkan masalah.
 Tidak ada standar tentang nilai simpul dan busur
Bingkai
 Bingkai adalah jaring semantik dengan properti
 Mewakili konsep umum atau entri khusus
 Bingkai mewakili objek sebagai set pasangan slot/pengisi
 Objek dapat berisi program dan juga data (jika diperlukan, jika ditambahkan, jika dihapus).
 Kegunaan frame terletak pada hirarki sistem frame dan pewarisan.
 Ini membuatnya mudah untuk membangun dan memanipulasi basis pengetahuan yang kompleks.
 Bingkai secara implisit terkait satu sama lain karena nilai slot dapat berupa bingkai lain
Ada tiga komponen bingkai:
(saya). Nama bingkai
(ii). Atribut (slot)
(aku aku aku). Nilai (Pengisi)
- Pengisi dapat menjadi tautan ke bingkai lain
Keuntungan

Model pengetahuan domain tercermin secara langsung
Mendukung penalaran default
Efisien
Mendukung pengetahuan prosedural
Kekurangan
- Kurangnya semantik
Batasan ekspresif

6.8 Kelas Sistem Pakar

6.6.1 Sistem pakar berbasis aturan

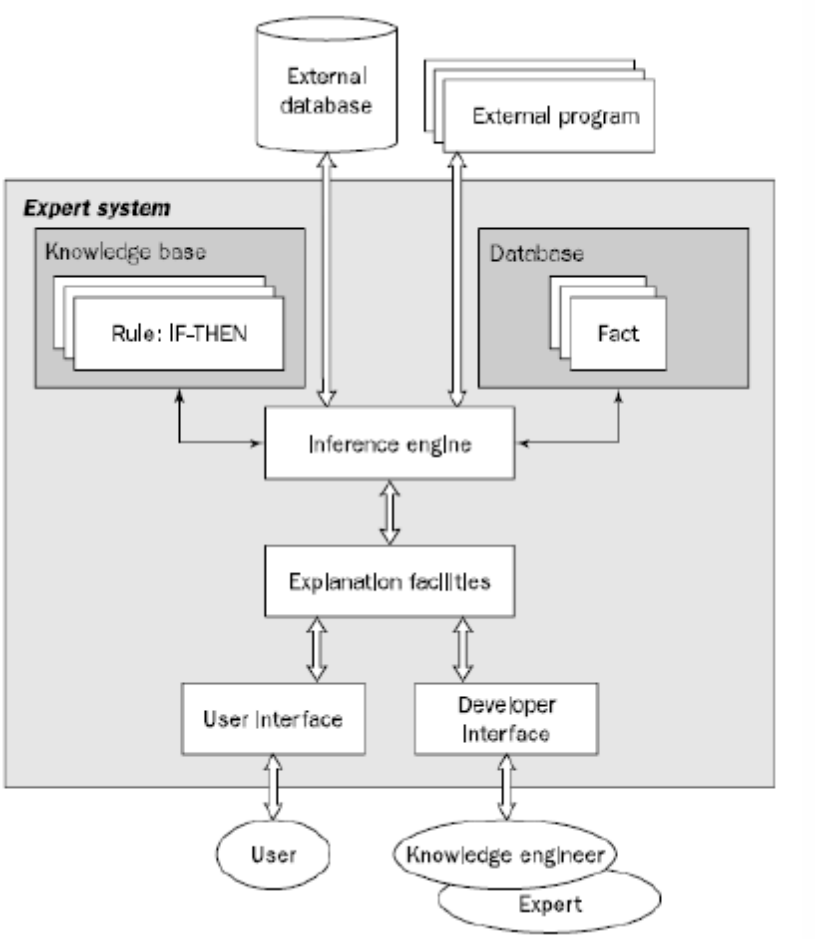
Sistem Pakar Berbasis Aturan: adalah jenis sistem pakar paling awal dan cara paling umum bagi para peneliti untuk membangun sistem pakar. Sistem pakar, yang dilengkapi dengan aturan produksi, telah diintegrasikan ke dalam berbagai bidang penelitian untuk membantu peneliti memecahkan semua jenis masalah dengan pengetahuan pra-input.

Sistem pakar berbasis aturan adalah sistem pakar yang bekerja sebagai sistem produksi di mana aturan mengkodekan pengetahuan pakar. Kebanyakan sistem pakar berbasis aturan. Ini adalah sistem pakar yang didasarkan pada seperangkat aturan yang akan diikuti oleh seorang pakar manusia dalam mendiagnosis suatu masalah. Contoh klasik dari sistem berbasis aturan adalah sistem pakar khusus domain yang menggunakan aturan untuk membuat deduksi atau pilihan. Misalnya, sistem pakar mungkin membantu dokter memilih diagnosis yang benar berdasarkan sekelompok gejala, atau memilih gerakan taktis untuk bermain game.

Proses dari pemilihan aturan untuk melakukan operasi biasanya kasus yang berisi beberapa kondisi masalah, mencari kondisi yang cocok dari database, dan menemukan aturan untuk kondisi yang sesuai. Ketika lebih dari satu kondisi cocok, Anda perlu mengurutkan strategi untuk memutuskan aturan mana yang akan digunakan terlebih dahulu. Setelah aturan dipilih, bagian operasi (hasil) aturan dijalankan. Pengoperasian strategi kontrol biasanya dilakukan dalam modul ("mesin inferensi"). Proses inferensi mesin inferensi dapat dibagi menjadi maju dan mundur.

Sistem Pakar Berbasis Aturan memiliki banyak keunggulan. Menggunakan pernyataan "kondisi - hasil" untuk mengungkapkan aturan pengetahuan sangat alami bagi manusia. Pada saat yang sama, dalam Sistem Pakar Berbasis Aturan, pengetahuan dan penalaran disimpan dan diproses secara terpisah, sesuai dengan kebiasaan sehari-hari masyarakat. Namun, beberapa kekurangan juga menghambat pengembangan lebih lanjut dari sistem pakar tersebut. Misalnya, ketika aturan cocok dengan suatu kondisi, ekspresi pernyataan harus ditulis secara ketat sesuai dengan pernyataan dalam database, bahkan jika perbedaan relatif halus ditolak karena persyaratan akurasi pencocokan. Selain itu, kecepatan Sistem Pakar Berbasis Aturan tidak dominan dibandingkan dengan jenis sistem pakar lainnya, karena seluruh aturan yang ditetapkan dalam database perlu dipindai saat aturan digunakan.

Struktur sistem pakar berbasis aturan ditunjukkan pada diagram di bawah ini. Antarmuka pengembang biasanya mencakup editor basis pengetahuan, alat bantu debugging, dan fasilitas input/output. Semua cangkang sistem pakar menyediakan editor teks sederhana untuk memasukkan dan memodifikasi aturan, dan untuk memeriksa format dan ejaan yang benar.



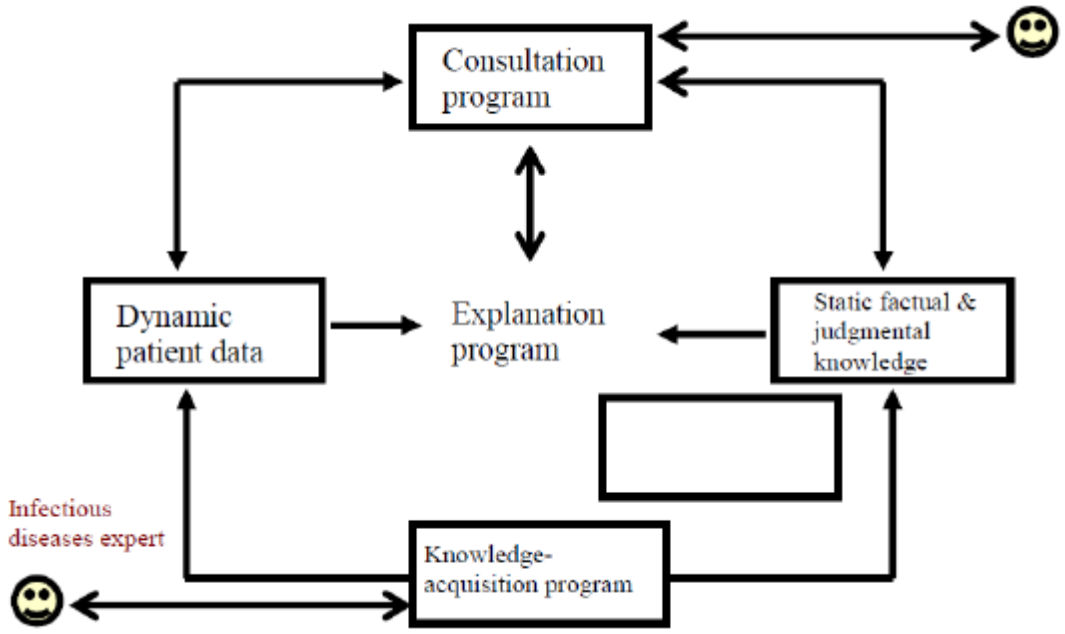
Gambar 6.7: menunjukkan struktur lengkap sistem pakar berbasis aturan

6.8.2 Mycin- Sebagai Contoh Sistem Pakar Berbasis Aturan

Ini adalah nama sistem pendukung keputusan yang dikembangkan oleh Universitas Stanford pada awal hingga pertengahan tahun tujuh puluhan, yang dibuat untuk membantu dokter dalam mendiagnosis penyakit menular. Sistem (juga dikenal sebagai "sistem pakar") akan mengajukan serangkaian pertanyaan yang dirancang untuk meniru pemikiran seorang ahli di bidang penyakit menular, dan dari tanggapan atas pertanyaan-pertanyaan ini memberikan daftar kemungkinan diagnosis, dengan probabilitas, serta merekomendasikan pengobatan (maka "dukungan keputusan"). Nama "MYCIN" sebenarnya berasal dari antibiotik, banyak di antaranya memiliki akhiran "-mycin".

Kerangka kerja untuk MYCIN berasal dari sistem pakar sebelumnya yang disebut DENDRAL, dibuat untuk menemukan senyawa kimia baru di bidang kimia organik (juga dikembangkan di Stanford).

Komponen Mycin



Gambar 6.8: Komponen sistem Mycin

– Khusus pasien
– Struktur hierarkis: Pasien, kultur, organisme
– <Objek, Atribut, Nilai> tiga kali lipat: <Org1, Identitas, Strep>
– CF yang digunakan untuk kepastian faktual <Org1, Identitas, Staph, 0,6>
Mycin adalah sistem pakar yang terdiri dari dua komponen utama:
1. Basis pengetahuan yang menyimpan informasi yang "diketahui" oleh sistem pakar, yang sebagian besar berasal dari informasi lain dalam basis pengetahuan.
2. Mesin inferensi untuk memperoleh pengetahuan dari pengetahuan yang sekarang dikenal dalam basis pengetahuan.
Pengetahuan yang diwakili oleh basis pengetahuan MYCIN direpresentasikan sebagai seperangkat aturan IF-THEN dengan faktor kepastian tertentu. Sebagai contoh:
JIKA infeksinya adalah bakteremia primer
DAN situs budaya adalah salah satu situs steril
DAN pintu masuk yang dicurigai adalah saluran cerna
MAKA ada bukti sugestif (0,7) bahwa infeksi itu adalah bakteri.

Manusia berinteraksi dengan MYCIN dengan menjawab serangkaian pertanyaan diagnostik yang mirip dengan apa yang mungkin ditanyakan dokter kepada pasien, serta meminta hasil tes yang relevan. MYCIN mengambil data ini sebagai input dan sampai pada serangkaian jawaban dengan probabilitas masing-masing, atau bercabang ke pertanyaan lain untuk mempersempit pencariannya. Para peneliti di Stanford menemukan MYCIN memiliki tingkat ketepatan perkiraan 65%, yang lebih baik daripada sebagian besar dokter yang bukan spesialis dalam mendiagnosis infeksi, dan hanya sedikit lebih buruk daripada dokter yang ahli di bidang itu (yang memiliki rata-rata ketepatan sekitar 80%).

Basis pengetahuan Mycin
Ini relatif kecil dibandingkan dengan yang digunakan oleh sebagian besar sistem berbasis aturan saat ini; itu ada di urutan ~ 500 aturan. Ilmu generasi aturan ini dikenal sebagai "rekayasa pengetahuan". MYCIN menggunakan modifikasi metode penalaran yang disebut "backward chaning" untuk mencari basis pengetahuannya. Modifikasi terjadi saat sistem berada pada tahap awal diagnosis, saat sistem mengajukan serangkaian pertanyaan umum untuk menyingkirkan pencarian yang tidak perlu di kemudian hari, seperti

memeriksa kehamilan jika pasiennya laki-laki. Hanya ketika masalahnya menjadi lebih jelas, MYCIN menggunakan rantai mundur penuh. Jika MYCIN memeriksa aturan dengan probabilitas 0,2 atau kurang, itu akan mengabaikan pencarian lebih lanjut pada aturan tertentu.

6.8.3 Mesin Inferensi

Inti dari MYCIN, mesin inferensinya, disebut EMYCIN ("MYCIN Esensial"). EMYCIN adalah kerangka kerja untuk MYCIN, sistem semi-terpisah yang dapat digunakan untuk membuat sistem pakar berbasis aturan lainnya untuk menghadapi masalah yang serupa dengan yang dihadapi MYCIN. Dalam beberapa kasus ini dapat dipengaruhi hanya dengan mengubah basis pengetahuan.

Fitur apa yang membuat sistem pakar berbasis aturan sangat menarik bagi para insinyur pengetahuan?

1- Representasi pengetahuan alam. Seorang ahli biasanya menjelaskan prosedur pemecahan masalah dengan ekspresi seperti ini: Dalam situasi ini dan itu, saya melakukan ini dan itu. Ekspresi ini dapat direpresentasikan secara alami sebagai aturan produksi IF-THEN.

2- Struktur seragam. Aturan produksi memiliki struktur IF-THEN yang seragam. Setiap aturan adalah bagian independen dari pengetahuan. Sintaks aturan produksi memungkinkan mereka untuk didokumentasikan sendiri.

3- Pemisahan pengetahuan dari pengolahannya. Struktur sistem pakar berbasis aturan menyediakan pemisahan yang efektif dari basis pengetahuan dari mesin inferensi. Hal ini memungkinkan untuk mengembangkan aplikasi yang berbeda menggunakan shell sistem pakar yang sama. Ini juga memungkinkan perluasan sistem pakar yang anggun dan mudah. Untuk membuat sistem lebih pintar, seorang insinyur pengetahuan hanya menambahkan beberapa aturan ke basis pengetahuan tanpa campur tangan dalam struktur kontrol.

4- Berurusan dengan pengetahuan yang tidak lengkap dan tidak pasti. Sebagian besar sistem pakar berbasis aturan mampu merepresentasikan dan menalar dengan pengetahuan yang tidak lengkap dan tidak pasti.

Sistem pakar berbasis aturan tidak bebas masalah?

Ada tiga kekurangan utama:

1- Hubungan buram antar aturan. Meskipun aturan produksi individu cenderung relatif sederhana dan didokumentasikan sendiri, interaksi logis mereka dalam kumpulan besar aturan mungkin buram. Sistem berbasis aturan membuat sulit untuk mengamati bagaimana aturan individu melayani strategi keseluruhan. Masalah ini terkait dengan kurangnya representasi pengetahuan hierarkis dalam sistem pakar berbasis aturan.

2- Strategi pencarian yang tidak efektif. Mesin inferensi menerapkan pencarian lengkap melalui semua aturan produksi selama setiap siklus. Sistem pakar dengan seperangkat aturan yang besar (lebih dari 100 aturan) bisa lambat, dan dengan demikian sistem berbasis aturan yang besar bisa jadi tidak cocok untuk aplikasi waktu nyata.

3- Ketidakmampuan untuk belajar. Secara umum, sistem pakar berbasis aturan tidak memiliki kemampuan untuk belajar dari pengalaman. Tidak seperti seorang ahli manusia, yang tahu kapan harus 'melanggar aturan', sistem pakar tidak dapat secara otomatis mengubah basis pengetahuannya, atau menyesuaikan aturan yang ada atau menambahkan yang baru. Insinyur pengetahuan masih bertanggung jawab untuk merevisi dan memelihara sistem.

6.9 Kerangka Sistem Pakar

Sistem pakar berbasis bingkai: Bingkai adalah struktur data dengan pengetahuan khas tentang objek atau konsep tertentu. Bingkai pertama kali diusulkan oleh Marvin Minsky pada 1970-an. Setiap frame memiliki nama sendiri dan satu set atribut yang terkait dengannya. Nama, berat badan, tinggi badan dan usia adalah slot dalam bingkai Orang. Model, prosesor, memori dan harga adalah slot dalam bingkai Komputer. Setiap atribut atau slot memiliki nilai yang melekat padanya. Bingkai menyediakan cara alami untuk representasi pengetahuan yang terstruktur dan ringkas. Bingkai menyediakan sarana untuk mengatur pengetahuan dalam slot untuk menggambarkan berbagai atribut dan karakteristik objek. Frame adalah aplikasi pemrograman berorientasi objek untuk sistem pakar. Konsep bingkai didefinisikan oleh kumpulan slot. Setiap slot

menggambarkan atribut atau operasi tertentu dari frame. Slot digunakan untuk menyimpan nilai. Sebuah slot mungkin berisi nilai default atau pointer ke frame lain, seperangkat aturan atau prosedur dimana nilai slot diperoleh. Bingkai mencakup informasi tentang cara menggunakan bingkai, informasi tentang harapan (yang mungkin salah), dan informasi tentang apa yang harus dilakukan jika harapan tidak dikonfirmasi, dan sebagainya. Kumpulan dari frame-frame tersebut harus diatur dalam sistem frame dimana frame-frame tersebut saling berhubungan. Sistem berbasis bingkai adalah sistem representasi pengetahuan yang menggunakan bingkai, sebagai sarana utama mereka untuk mewakili pengetahuan domain. Pengetahuan diatur dalam bentuk potongan yang disebut bingkai. Bingkai ini seharusnya menangkap esensi konsep atau situasi stereotip. Misalnya, berada di ruang tamu atau pergi makan malam, dengan mengelompokkan semua informasi yang relevan untuk situasi ini bersama-sama.

Sistem Pakar Berbasis Bingkai dikaitkan dengan objek yang menarik dan penalaran terdiri dari konfirmasi harapan untuk nilai slot. Sistem seperti itu sering kali menyertakan aturan juga. Dalam sistem berbasis bingkai, mesin inferensi juga mencari tujuan, atau dalam istilah lain untuk atribut yang ditentukan, hingga nilainya diperoleh. Dalam sistem pakar berbasis aturan, tujuannya ditentukan untuk basis aturan. Dalam sistem berbasis bingkai, aturan memainkan peran tambahan. Bingkai mewakili di sini sumber utama pengetahuan, dan baik metode maupun iblis digunakan untuk menambahkan tindakan ke bingkai. Jadi, kita mungkin berharap bahwa tujuan dalam sistem berbasis bingkai dapat ditetapkan baik dalam metode atau dalam iblis. Demon memiliki struktur IF-THEN. Itu dieksekusi setiap kali atribut dalam pernyataan IF iblis mengubah nilainya. Dalam pengertian ini, setan dan metode sangat mirip, dan kedua istilah tersebut sering digunakan sebagai sinonim. Namun, metode lebih tepat jika kita perlu menulis prosedur yang rumit. Demons, di sisi lain, biasanya terbatas pada pernyataan IF-THEN.

Struktur sistem pakar berbasis kerangka kerja sebagai "bingkai". Kerangka kerja berisi nama konsep, slot label atribut/fitur utama, dan nilai yang mungkin dari setiap atribut, atau proses menangkap informasi prosedural tentang konsep tersebut. Ketika contoh tertentu dari konsep ditemukan, nilai eigen yang relevan dari contoh tersebut dimasukkan ke dalam kerangka kerja, yang disebut contoh. Kerangka tersebut dapat digunakan untuk penalaran. Kerangka tersebut berisi informasi multi-aspek dari suatu konsep yang dapat digunakan bahkan jika informasi itu sendiri tidak diamati. Misalnya, apakah kita dapat melihat akarnya atau tidak, karena "pohon" berisi "akar" dari label, sehingga kita dapat berpikir bahwa pohon itu memiliki akar. Saat mencari kerangka kerja untuk menggambarkan contoh saat ini, sering kali tidak cocok dengan situasi. Contoh spesifik akan sesuai dengan beberapa fragmen bingkai yang digunakan untuk mencocokkan bingkai kandidat. Kerangka kerja biasanya digunakan ketika sebagian besar struktur bingkai dapat dicocokkan, karena setiap bingkai berisi informasi yang memungkinkan ketidakcocokan atribut (fitur) untuk meningkatkan toleransi kesalahan bingkai. Selain itu, kita juga dapat menyimpan kerangka proses pencarian untuk mengoptimalkan arah pengujian hingga menemukan kerangka kerja yang paling sesuai.

Sistem Pakar Berbasis Kerangka menggunakan kerangka kerja dalam database untuk menangani isu-isu spesifik dari input dan output informasi baru melalui mesin inferensi. Kerangka kerja mewakili konsep "kelas", dan kerangka kerja mungkin merupakan "subkelas" dari kerangka kerja lain, seperti kerangka kerja "manusia" adalah subset dari kerangka kerja "manusia". Beberapa hubungan subkelas dan kelas adalah tipe semantik, seperti "Saya" + "New York", "New Jersey", "Los Angeles".

Kerangka subkelas mewarisi semua properti kerangka kelas induknya, menghilangkan kebutuhan untuk memasukkan kembali properti proses. Namun, kita harus memperhatikan beberapa keadaan khusus: beberapa subclass dan kelas ayah mereka mungkin dalam perbedaan properti umum. Ketika sebuah bingkai dimiliki oleh beberapa bingkai pada saat yang sama, ia juga mewarisi properti dari semua frame ini.

6.9.1 Istilah Terkait dengan Sistem Pakar berbasis frame

- Frame : Struktur data dengan pengetahuan khas tentang objek tertentu.
- Slot: Sebuah komponen frame dalam sistem berbasis frame yang menggambarkan atribut tertentu dari frame.
- Atribut: Sebuah datum yang terkait dengan entitas.

6.9.2 Frame Sebagai Teknik Representasi Pengetahuan

- Nama frame
- Class-Frame: Sebuah frame yang mewakili sebuah kelas.
- Instance-Frame: Sebuah frame yang mewakili sebuah instance.
- Facet: Sebuah komponen slot dalam sebuah frame.

6.9.3 Slot frame Dapat Mencakup:

- Nama frame
- Hirarki frame : frame induk
- Nilai atribut. Ini bisa berupa angka, nilai Boolean, atau simbol (karakter).
- Nilai default untuk atribut.
- Rentang nilai yang valid untuk atribut.

Sebuah subprogram yang dijalankan ketika nilai slot diubah atau diakses. Subprogram ini disebut setan di dunia AI. (Ini tidak sama dengan setan, atau daemon, dari dunia UNIX).
Demons dapat dipicu ketika:

- Nilai baru ditambahkan ke slot.
- Nilai dihapus dari slot.
- Nilai dalam slot diganti dengan nilai baru.
- Nilai diminta dari slot kosong (yang tidak berisi nilai).

6.9.4 the demons adalah segi:

Istilah "frame" digunakan untuk merujuk baik ke bingkai kelas, yang merupakan templat abstrak, dan ke frame instan, yang merupakan analog konkret dari entitas tertentu.

6.9.5 Warisan pada Pakar Berbasis frame

Frame
Dalam konteks frame, pewarisan mengacu pada:

- Warisan oleh kerangka kelas dari karakteristik (slot) dari semua superkelasnya.
- Warisan oleh instance-frame dari karakteristik class-frame-nya.

6.10 Aplilasi Penelitian yang berhubungan dengan Pakar

1. Sistem Pakar Berbasis Web Dengan Metode Forward Chaining Dalam Mendiagnosa Dini Penyakit Tuberkulosis Di Jawa Timur.
   Pada penelitian ini, peneliti mebangun sebuah web yang bertujuan untuk mendiagnosa penyakit tuberkolosi di jawa timur. Data yang digunkan berasal dari TB *Care* Aisyiyah Jawa Timur yang beralamatkan di Jl. Kertomenanggal IV No.1 Gayungan, Kota Surabaya, Jawa Timur,

KINETIK, Vol.1, No.3, 2016, November Hal. 147-154
ISSN : 2503-2259,
E-ISSN : 2503-2267

Sistem Pakar Berbasis Web Dengan Metode Forward Chaining Dalam Mendiagnosis Dini Penyakit Tuberkulosis di JawaTimur

Windah Supartini*[1], Hindarto[2]
[1, 2]Universitas Muhammadiyah Sidoarjo
windaumsida@gmail.com*[1], hindarto@umsida.ac.id[2]

Abstrak

Tuberkulosis adalah suatu penyakit menular berbahaya yang disebabkan oleh kelompok Mycobacterium, yaitu Microbacterium Tuberkulosis. Setiap pasien Tuberkulosis dapat menularkan penyakitnya pada orang lain yang berada di sekelilingnya dan atau yang berhubungan erat dengannya. Karena masih banyak orang yang tidak mengetahui gejala-gejala penyakit suatu sistem pakar mendiagnosis secara dini penyakit tuberkulosis menggunakan metode forward chaining berbasis web, dapat dikenali dengan melihat gejala-gejala dengan mendeteksi penyakit sejak dini, dilakukan pencegahan terhadap penyakit tuberkulosis. Diagnosis sistem pakar, memiliki nilai keakuratan 93,333 % dan nilai eror 6,667 % untuk uji coba pada 15 pasien. Sehingga dapat disimpulkan bahwa sistem pakar cukup layak untuk digunakan oleh pasien dalam mendiagnosis dini pada penyakit tuberkulosis.

***Kata Kunci:** Tuberkulosis, Forward Chaining, Sistem Pakar, Web*

Abstract

Tuberculosis is a contagious disease caused by Mycobacterium family; Mycobacterium Tuberkulosis. Every patient of tuberculosis could possibly transmit this disease to people around him, or to people that closely related to him. Because there are still many people do not really know about the symptoms, an expert system is developed to diagnose tuberculosis by using web-based forward chaining method; by recognizing the disease earlier, the prevention could be conducted appropriately. Diagnoses expert system has 93.333% accuracy and 6.667% errors as experimented to 15 patients. Therefore, it could be concluded that expert system is adequate to be used by patients to diagnose tuberculosis disease.

***Keywords:** Tuberculosis, Forward Chaining, Expert System, Web*

1. Pendahuluan

Tuberkulosis adalah suatu penyakit menular berbahaya yang disebabkan oleh dari kelompok Mycobacterium, yaitu Mycobacterium Tuberkulosis. Setiap pasien Tuberkulosis dapat menularkan penyakitnya pada orang lain yang berada di sekelilingnya dan berhubungan erat dengannya [1].

Jumlah pasien TB di Indonesia merupakan jumlah ketiga di dunia setelah India dan Cina. Dari hasil persentase antara lain India 21.1%, China 14.3%, Indonesia 5.8%, Nigeria 4.9%, Ethiopia 3.3%, Pakistan 3.2%, dan lainnya 15.9%. Setiap tahun diperkirakan terjadi 583.000 pasien baru TB setiap tahun, diperkirakan 140.000 orang meninggal karena TB. Setiap menit muncul satu orang pasien TB Paru Baru, setiap 2 menit muncul satu orang penderita TB Paru yang menular, setiap 1 menit satu orang meninggal akibat TB. Setiap hari sekitar 400 orang meninggal karena TB yang sebanding dengan jumlah korban penumpang pesawat jenis boeing seri 747 yang jatuh setiap hari. TB merupakan penyebab kematian nomor satu dari golongan penyakit infeksi, dan nomor tiga penyebab kematian pada semua kelompok usia setelah penyakit jantung dan penyakit saluran pernafasan. Saat ini, program penanggulangan TB dengan Strategi DOTS (*Directly Observed Treament Short-course*) atau disebut dengan pengawasan langsung jangka pendek belum dapat menjangkau seluruh Puskesmas, Rumah Sakit Pemerintah atau Swasta, serta unit pelayanan kesehatan lainnya [1].

Jumlah penderita TB dapat ditekan dengan adanya tindakan pendeteksian dini gejala penyakit TB. Oleh karena hal tersebut, dibutuhkan suatu aplikasi *system* pakar agar mudah digunakan oleh orang awam mendiagnosis penyakit Tuberkulosis.

Makalah dikirim - 2016; Revisi 1 Oktober 2016; Diterima 30 November 2016

2. Metode Penelitian

Dalam penelitian ini, metode yang dilakukan, adalah memeriksa berkas pasien Tuberkulosis dan melakukan wawancara di *sub recipient* TB Care Aisyiyah Jawa Timur yang beralamatkan di Jl. Kertomenanggal IV No.1 Gayungan, Kota Surabaya, Jawa Timur, 60234 dengan dokter yang direkomendasikan untuk penelitian dari Rumah Sakit/Puskesmas Program Penanggulangan TB Care Aisyiyah, yaitu dr. Daniek Suryaningdiah sebagai dokter yang menangani program penyakit Tuberkulosis.

2.1 Bahan Penelitian

Bahan yang digunakan untuk penelitian ini adalah data-data dari dalam buku Pedoman Nasional "Pengendalian Tuberkulosis" oleh (Kementerian Kesehatan Republik Indonesia Direktorat Jendral Pengendalian penyakit dan Penyehatan Lingkungan, 2014), serta data dari *sub recipient* TB Aisyiyah Jatim dan setiap *sub recipient* TB Aisyiyah Surabaya, "*Artificial Inteligance*" oleh Suyanto, ST., MSC. Serta buku-buku dan sumber lain yang menunjang pembuatan Perancangan Sistem Pakar Berbasis Web dengan metode Forward Chaining dalam upaya mendiagnosis dini penyakit Tuberkulosis di Jawa Timur.

2.2 Teknik Pengumpulan Data

Adapun teknik pengumpulan data yang digunakan untuk penelitian sistem pakar diagnosis penyakit Tuberkulosis berbasis web dengan metode Forward Chaining adalah sebagai berikut:

2.2.1 Studi pustaka

Dalam penelitian ini, dilakukan pencarian dan pembelajaran dari berbagai macam sumber pustaka. Di antaranya data program penanggulangan TB di Jawa Timur oleh SR TB Aisyiyah Jawa Timur Informasi dari dokter spesialis penanganan penyakit Tuberkulosis, buku-buku, jurnal, dan website yang berkaitan dengan perancangan sistem pakar berbasis web dengan metode Forward Chaining dalam upaya mendiagnosis dini penyakit Tuberkulosis di Jawa Timur.

2.2.2 Observasi

Observasi adalah mengadakan penelitian dan analisis secara langsung terhadap data yang akan diteliti, yaitu mengamati ciri-ciri dan keluhan orang yang diduga TB (Suspek) dengan cara tes uji coba penggunaan sistem pakar sampai dapat diketahui hasilnya negatif/positif dan jika positif maka dapat diketahui Klasifikasinya.

2.2.3 Wawancara

Dalam pengembangan sistem pakar, dilakukan tanya jawab secara langsung kepada dokter spesialis penyakit Tuberkulosis untuk mendapatkan informasi yang dibutuhkan dalam proses perancangan sistem pakar berbasis web dengan metode Forward Chaining dalam upaya mendiagnosis dini penyakit Tuberkulosis di Jawa Timur.

2.3 Sistem Pakar

Sistem pakar merupakan cabang dari AI (*Artificial Inteligent*) yang membuat eksistensi untuk spesialisasi pengetahuan guna memecahkan suatu permasalahan pada *Human Expert*. *Human Expert* merupakan seseorang ahli dalam suatu bidang ilmu pengetahuan tertentu, berarti *expert* memiliki suatu permasalahan yang tidak dapat dipecahkan oleh orang lain secara efisien.

2.3.1 Forward Chaining

Forward Chaining merupakan fakta untuk mendapatkan kesimpulan (*conclusion*) dari fakta tersebut. [2] Penalaran ini berdasarkan fakta yang ada (*data driven*), metode ini adalah kebalikan metode Backward Chaining, dimana metode ini dijalankan dengan mengumpulkan fakta-fakta yang ada untuk menarik kesimpulan. Dengan kata lain, prosesnya dimulai dari *facts* (fakta-fakta yang ada) melalui proses *interface fact* (penalaran fakta-fakta) menuju suatu *goal* (suatu tujuan). Metode ini juga disebut menggunakan aturan IF–THEN dimana premise (IF) menuju *conclusion* (THEN) atau dapat juga dituliskan sebagai berikut:

THEN (konklusi)

Ada dua pendapat mengenai pelaksanaan metode ini. Pertama dengan cara membawa seluruh data yang didapat ke sistem pakar. Kedua dengan membawa bagian-bagian penting saja dari data yang didapat ke sistem pakar. Cara pertama lebih baik digunakan jika sistem pakar terhubung dengan proses otomatis dan penerima seluruh data dari *database*. Cara kedua menghemat waktu serta biaya dengan mengurangi data dan mengambil data yang dianggap perlu. Sebagai contoh, seperti kasus pada kedua metode di atas, maka berdasarkan metode ini langah-langkah yang diambil:

R1 : IF A and C, THEN B
R2 : IF D and C, THEN F
R3 : IF B and E, THEN F
R4 : IF B, THEN C
R5 : IF F, THEN G

Kedua jenis strategi ini akan mengarah pada suatu kesimpulan. Namun, efisiensinya tergantung dari kondisi masalah yang dihadapi, jika suatu masalah memiliki premis jumlahnya lebih sedikit dibanding *conclusion*, maka strategi yang akan ditawarkan Backward Chaining [3].

2.4 Perangkat Perancangan Sistem

2.4.1 Basis data (*Database*)

Basis data atau *database* adalah kumpulan data yang disimpan secara sistematik di dalam komputer. Basisdata juga dapat diolah atau dimanipulasi menggunakan perangkat lunak atau program aplikasi untuk menghasilkan suatu informasi. Pendefinisian basis data meliputi spesifikasi berupa tipe data, struktur data, dan juga batasan-batasan data yang akan disimpan [4].

2.4.2 Tabel Relasi

Tabel relasi merupakan hubungan yang terjadi pada suatu tabel dengan tabel lain, yang berfungsi untuk mengatur operasi suatu *database*. Hubungan yang dibentuk dapat mencangkup tiga macam hubungan, yaitu [5]:

1. One-To-One (1-1)
2. One- to-Many (1-N)
3. Many-To-Many (N-M)

2.4.3 *Flowchart*

Flowchart atau diagram alir merupakan sebuah diagram dengan simbol-simbol grafis yang menyatakan aliran algoritma atau proses menampilkan langkah-langkah yang disimbolkan dalam bentuk kotak, beserta urutannya dengan menghubungkan masing-masing langkah tersebut menggunakan tanda panah. Diagram ini dapat memberi solusi langkah demi langkah untuk penyelesaian masalah yang ada dalam proses atau algoritma tersebut [3].

2.4.4 Data *Flow* Diagram

Data *Flow* Diagram (DFD) adalah diagram yang menggunakan notasi-notasi untuk menggambarkan arus dari data sistem, yang mana penggunaannya sangat membantu memahami sistem secara logika, terstruktur, dan jelas. DFD merupakan alat bantu dalam menggambarkan atau menjelaskan proses kerja suatu sistem informasi maupun sistem aplikasi [4].

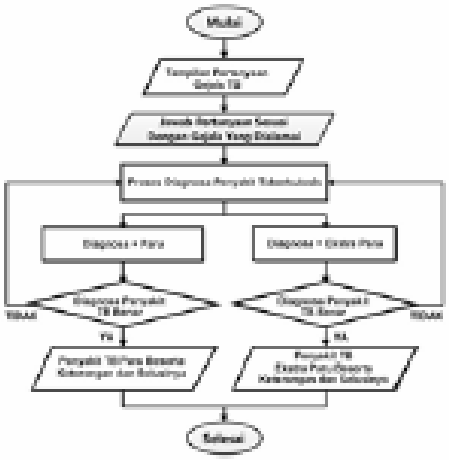
Pada DFD terdapat 3 level, yaitu [6]:

1. Diagram Konteks, merupakan level tertinggi dari DFD, yang mana menggambarkan tentang seluruh *input* dan *output* dari suatu sistem.
2. Diagram Nol, Merupakan pemecahan dari diagram konteks. Dalam diagram nol ini memuat penyimpanan data dan proses dari suatu sistem.
3. Diagram Rinci, merupakan uraian proses dari diagram nol atau diagram di level atasnya dalam suatu sistem.

2.5 Mesin Inferensi

Dalam sebuah sistem pakar dibutuhkan sebuah mesin sistem inferensi yang akan menjadi pengendali dari sebuah sistem pakar. Mesin inferensi merupakan fungsi berpikir dan

pola penalaran sistem yang digunakan oleh sistem pakar. Mekanisme ini akan menganalisis suatu masalah dan selanjutnya akan mencari jawaban serta kesimpulan yang terbaik. Teknik dalam mesin inferensi sistem pakar pada Gambar 1 adalah metode Forward Chaining.



Gambar 1. Mesin Inferensi Sistem Pakar Mendiagnosis Penyakit Tuberkulosis

2.5.1 Analisis

Basis Pengetahuan Gejala Penyakit Tuberkulosis, terpapar pada Tabel 1.

Tabel 1. Tabel Relasi Gejala pada Penyakit Tuberkulosis

	Penyakit		
Kode	P001	P002	P003
G001	X	X	
G002	X	X	
G003	X		
G004	X		
G005	X		
G006	X		
G007	X		
G008	X		
G009	X		
G010	X		
G011		X	
G012		X	
G013		X	
G014		X	
G015			X
G016			X
G017			X
G018			X
G019			X
G020			X
G021			X
G022			X
G023			X
G024			X
G025			X
G026			X

3. Hasil Penelitian dan Pembahasan
3.1 Form antarmuka pemakai

Gambar 2. Tampilan Utama Sistem Pakar

Tampilan Gambar 2 merupakan tampilan utama sistem pakar yang akan dijumpai oleh *user*. Tampilan ini berisi tentang sambutan dan berita terkini mengenai penyakit Tuberkulosis yang ditujukan untuk *user* sistem pakar.

Gambar 3. Tampilan Menu User

Tombol registrasi Gambar 3 digunakan untuk pendaftaran pasien. Serta terdapat form *login user* digunakan untuk *login* pada tampilan utama menu pasien.

Gambar 4. Tampilan Utama Menu Pasien

Gambar 4 adalah tampilan menu pasien yang sudah *login*, muncul menu pertanyaan untuk mengetahui diagnosis pasien dan selanjutnya menu hasil riwayat diagnosis.

Gambar 5. Tampilan Form Diagnosis Penyakit Tuberkulosis

Pada Gambar 5, terdapat tampilan form diagnosis penyakit Tuberkulosis, terdapat beberapa pertanyaan seputar gejala Tuberkulosis secara umum dan spesifik harus diisi oleh pasien untuk mengetahui jenis penyakit Tuberkulosis yang diderita oleh pasien saat ini.

Gambar 6. Tampilan Kesimpulan Hasil Diagnosis

Pada tampilan Gambar 6, pasien dapat mengetahui gejala-gejala yang sudah dialami termasuk dapat diketahui jenis penyakit yang diderita oleh pasien saat ini.

Gambar 7. Tampilan Hasil Diagnosis Penyakit Tuberkulosis

Pada tampilan Gambar 7 adalah hasil diagnosis penyakit Tuberkulosis, pasien dapat mengetahui jenis penyakit Tuberkulosis yang sedang diderita saat ini, serta mendapatkan

keterangan dan solusi penyakit Tuberkulosis tersebut. Pasien juga dapat mencetak hasil diagnosis.

3.2 Perbandingan Hasil Diagnosis Sistem Pakar dengan Diagnosis Dokter

Tabel 2, 3, dan 4 adalah perbandingan hasil diagnosis penyakit Tuberkulosis menggunakan sistem Pakar dengan hasil diagnosis dokter yang sesungguhnya.

1. Pasien 1

Nama Pasien : Rukanah
Umur : 53 Tahun

Gejala: Batuk berdahak selama 2 minggu atau lebih, batuk dengan dahak bercampur darah, batuk disertai darah, sesak nafas dan nyeri dada, badan lemas, nafsu makan menurun, berat badan menurun.

Tabel 2. Perbandingan Hasil Diagnosis Pasien 1

Diagnosis Sistem Pakar	Diagnosis Dokter
Tuberkulosis Paru (positif)	Tuberkulosis Paru dengan Hasil Positif

2. Pasien 2

Nama Pasien : Musinem
Umur : 40 Tahun

Gejala: Batuk berdahak selama 2 minggu atau lebih, batuk dengan dahak bercampur darah, batuk disertai darah, sesak nafas dan nyeri dada, badan lemas, nafsu makan menurun, berat badan menurun, malaise, berkeringat malam hari tanpa kegiatan fisik, demam meriang lebih dari 1 bulan.

Tabel 3. Perbandingan Hasil Diagnosis Pasien 2

Diagnosis Sistem Pakar	Diagnosis Dokter
Tuberkulosis Paru (positif)	Tuberkulosis Paru dengan Hasil Positif

Tabel 4. Perbandingan Hasil Diagnosis Pasien 15

Diagnosis Sistem Pakar	Diagnosis Dokter
Tuberkulosis Paru (positif)	Tuberkulosis Paru dengan Hasil Positif

Berdasarkan 15 perbandingan hasil diagnosis penyakit Tuberkulosis menggunakan sistem pakar dengan hasil diagnosis dokter di atas, terdapat 1 hasil diagnosis pasien menggunakan sistem pakar yang tidak sesuai dengan hasil diagnosis dokter, yaitu pada pasien 10. Maka dari itu, sistem pakar mendiagnosis secara dini pada penyakit Tuberkulosis mempunyai nilai error sesuai dengan Persamaan 1.

$$x = \frac{a}{b} x\ 100\% \tag{1}$$

$$= \frac{1}{15} x\ 100\ \% = 6{,}667\ \%$$

Keterangan:
Angka 1 = Jumlah hasil nilai error (hasil diagnosis sistem pakar yang tidak sama dengan hasil diagnosis dokter).
Angka 15 = Semua jumlah hasil diagnosis.

Sedangkan nilai keakuratan dari sistem pakar mendiagnosis secara dini pada penyakit Tuberkulosis dihitung menggunakan Persamaan 2.

$$x = \frac{a}{b} x\ 100\% \quad (2)$$

$$= \frac{14}{15} x\ 100\ \% = 93{,}333\ \%$$

Keterangan:
Angka 14 = Jumlah hasil nilai yang akurat (hasil diagnosis sistem pakar yang sama dengan hasil diagnosis dokter).
Angka 15 = Semua jumlah hasil diagnosis.

4. Kesimpulan

Dari hasil perancangan, pembuatan, pengimplementasian, serta pengujian aplikasi sistem pakar mendiagnosis secara dini pada penyakit Tuberkulosis menggunakan metode Forward Chaining berbasis web, dapat diperoleh berbagai kesimpulan dan saran untuk perkembangan program aplikasi sistem pakar mendiagnosis secara dini pada penyakit Tuberkulosis ke arah yang lebih baik.

1. Sistem pakar mendiagnosis secara dini pada penyakit Tuberkulosis menggunakan metode Forward Chaining berbasis web ini cukup membantu untuk mendiagnosis penyakit Tuberkulosis berdasarkan gejala-gejala yang dikeluhkan oleh pasien.
2. Hasil diagnosis pakar dan user dari sistem pakar mendiagnosis secara dini pada penyakit Tuberkulosis menunjukkan bahwa hasil diagnosis yang dialami pasien menunjukkan sesuai dengan yang telah di diagnosis oleh dokter penyakit Tuberkulosis.

6. Daftar Notasi

x = % error
a = data pasien yang tidak sesuai dengan diagnosis
b = banyak data pasien yang diagnosis

Referensi

[1] Tim Pengembangan Modul. Pelatihan penanggulangan Tuberkulosis bagi Kader Komunitas. Penerbit Recipient TB Alsylyah. Jakarta. 2009.
[2] Winarko, Edi. Perancangan Database Dengan Power Desaigner 6.32. Prestasi Pustakarya. Jakarta. 2006.
[3] Purwono, Edi. Sistem Analisis, ANDI. Yogyakarta. 2007.
[4] Azzolini, John. Introduction to System Engineering Practices, Yogyakarta: ANDI Offset. 2004.
[5] Ignizio, J.P. Introduction To Expert System: The Development and Implementation Of Rule-Based Expert System. McGraw-Hill International Editions. 1991.
[6] Giinratano, Riley. Expert Sistem: Principles and Programming, PWS Publising Company, Boston. 1994.

Soal :
Sistem Pakar Diagnosa Penyakit yang disebabkan oleh Nyamuk
Basis Pengetahuan :
Adapun Penyakit yang disebabkan oleh Nyamuk antara lain:

1. Demam Berdarah Denque
2. Demam Penyakit Kuning
3. Chikungunya
4. Encephalitis
5. Malaria

Beberapa gejala yang mungkin muncul dari kelima jenis penyakit tersebut adalah sebagai berikut:

- Demam
- Merasa Kedinginan
- Tubuh Terasa Sakit
- Sakit Kepala
- tenggorokan sakit saat menelan
- Badan Terasa Lemas dan Lemah
- Muncul Bintik-bintik berwarna Merah
- panas tubuh tinggi
- otot terasa nyeri
- nafsu makan menurun
- merasa mual-mual
- denyut nadi terasa lemah
- merasakan ngilu
- merasakan persendian membengkak
- stamina terasa menurun
- nyeri pada setiap persendian
- merasakan ingin muntah
- penglihatan terganggu bila melihat cahaya
- leher dan punggung terasa kaku
- sering merasa mengantuk
- mudah terangsang

Dari gejala-gejala tersebut kita dapat membuat aturan sebagai berikut:

1. Jika gejala yang timbul adalah **1,2, 3,4,5,6**, dan **7** maka dia menderita **Demam Berdarah Denque**
2. Jika gejala yang timbul **1,4,8,9,10,11**,dan **12** maka dia menderita **Demam Penyakit Kuning**
3. Jika gejala yang timbul adalah **1,11,13,14,15,16**, dan **17** maka dia menderita **Chikungunya**
4. Jika gejala yang timbul **1,4,17,18,19,20**, dan **21** maka dia menderita **Encephalitis**
5. Jika gejala yang timbul **1,2,4,6**, dan **8** maka dia menderita **Malaria**

Buat Pertanyaan :

ID	solusi_dan_pertanyaan	bila_benar	bila_salah	mulai	selesai
1	Apakah Anda merasakan demam tinggi ?	2	27	Y	N
2	Apakah Anda merasa kedinginan ?	3	4	N	N
3	Apakah Anda merasakan tubuh anda terasa sakit ?	4	5	N	N
4	Apakah Anda merasakan sakit kepala ?	5	11	N	N
5	Apakah Anda merasakan Tenggorokan sakit saat menel...	6	8	N	N
6	Apakah Anda merasakan badan lemas dan lemah ?	7	15	N	N
7	Apakah pada tubuh anda muncul Bintik-bintik berwar...	22	13	N	N
8	Apakah anda merasakan panas tubuh tinggi ?	26	0	N	N
9	Apakah tenggorokan Anda sakit bila menelan ?	10	10	N	N
10	Apakah anda merasakan otot anda terasa nyeri ?	11	0	N	N
11	Apakah nafsu makan anda menurun ?	12	13	N	N
12	Apakah anda merasa mual-mual ?	23	16	N	N
13	Apakah denyut nadi Anda terasa lemah ?	14	14	N	N
14	Apakah tubuh anda terasa ngilu ?	15	6	N	N
15	Apakah Anda merasakan persendian Anda membengkak ?	16	20	N	N
16	Apakah anda merasakan stamina menurun ?	17	14	N	N
17	Apakah Anda merasakan nyeri pada persendian ?	24	18	N	N
18	Apakah Anda merasakan ingin muntah ?	19	0	N	N
19	Apakah Anda merasakan leher dan punggung terasa ka...	20	0	N	N
20	Apakah Anda sering merasa ngantuk ?	21	0	N	N
21	Apakah Anda mudah terangsang ?	25	0	N	N
22	Anda menderita PENYAKIT DEMAM BERDARAH	0	0	N	Y
23	Anda menderita PENYAKIT DEMAM PENYAKIT KUNING	0	0	N	Y
24	Anda menderita PENYAKIT CIKUNGUNYA	0	0	N	Y
25	Anda menderita PENYAKIT ENCEPHALITIS	0	0	N	Y
26	Anda menderita PENYAKIT MALARIA	0	0	N	Y
0	MAAF UNTUK SEMENTARA, SISTEM INI BELUM DAPAT MENDI...	0	0	N	Y
27	Anda hanya menderita DEMAM BIASA	0	0	N	Y

Dari data diatas buatkan Mesin Inferensinya, pilih salah satu cara atau metode dibawah ini

1. Forward Chainning
2. Backward Chainning
3. Atau memakai Demster Shafer
4. dll

BAB 7
Logika Fuzzy

7.1 Pendahuluan

Di sebagian besar aplikasi saat ini, fuzzy logika memungkinkan banyak jenis desainer dan operator pengetahuan kualitatif dalam otomatisasi sistem untuk diperhitungkan. Logika fuzzy mulai menarik minat media di awal tahun sembilan puluhan. yang banyak aplikasi dalam listrik dan elektronik peralatan rumah tangga, khususnya di Jepang, terutama bertanggung jawab untuk kepentingan tersebut. Mesin cuci tidak memerlukan penyesuaian, camcorder dengan gambar Steadyshot (TM) stabilisasi dan banyak inovasi lain yang dibawa istilah "logika kabur" menjadi perhatian luas publik. Dalam industri mobil, perpindahan gigi otomatis, kontrol injeksi dan anti-rattle dan udara pengkondisian dapat dioptimalkan berkat fuzzy logika.
Dalam proses produksi terus menerus dan batch, serta dalam sistem otomasi (yang merupakan
subjek Teknik Cahier ini), aplikasi juga telah meningkat. Logika fuzzy telah berkembang di daerah ini karena pada dasarnya pragmatis, pendekatan yang efektif dan generik. Ini memungkinkan sistematisasi pengetahuan empiris dan yang karenanya sulit dikendalikan. teori kabur set menawarkan metode yang cocok yang mudah untuk implementasikan dalam aplikasi waktu nyata, dan aktifkan pengetahuan desainer dan operator untuk menjadi ditranskripsikan ke dalam sistem kontrol dinamis. Ini membuat logika fuzzy mampu menangani otomatisasi prosedur seperti startup dan pengaturan parameter, yang beberapa pendekatannya adalah sebelumnya tersedia. Teknik Cahier ini menjelaskan logika fuzzy dan penerapannya pada proses produksi.

7.1.1 Sejarah logika fuzzy

Penampilan logika fuzzy
Istilah "set fuzzy" pertama kali muncul pada tahun 1965 ketika profesor Lotfi A. Zadeh dari universitas Berkeley, AS, menerbitkan makalah berjudul "Set kabur". Sejak itu dia telah mencapai banyak terobosan teoretis utama di bidang ini dan telah dengan cepat bergabung dengan banyak penelitian pekerja mengembangkan karya teoritis.
Aplikasi awal
Pada saat yang sama, beberapa peneliti mengubah perhatian pada resolusi dengan logika fuzzy dari masalah yang dianggap sulit. Pada tahun 1975 profesor Mamdani dari London mengembangkan strategi untuk pengendalian proses dan menerbitkan hasil yang menggembirakan yang diperolehnya untuk pengendalian motor uap. Pada tahun 1978 orang Denmark perusahaan, F.L. Smidth, mencapai kontrol a tempat pembakaran semen. Ini adalah industri asli pertama aplikasi logika fuzzy.
Boom
Logika fuzzy mengalami ledakan yang nyata dalam Jepang di mana penelitian tidak hanya teoretis tetapi juga sangat berorientasi pada aplikasi. Pada akhir logika fuzzy tahun delapan puluhan telah berkembang pesat cara, dan produk konsumen seperti mencuci mesin, kamera, dan camcorder dengan menyebutkan "logika kabur" terlalu banyak untuk menjadi terhitung. Aplikasi industri seperti: pengolahan air, derek kontainer pelabuhan, bawah tanah dan ventilasi/AC sistem mulai menggunakan logika fuzzy juga. Akhirnya, aplikasi yang dikembangkan di bidang lain seperti: seperti keuangan dan diagnosis medis. Dari tahun 1990 dan seterusnya, banyak aplikasi mulai muncul dalam jumlah besar di Jerman, juga sebagai, pada tingkat lebih rendah, di Amerika Serikat.

7.1.2 Nilai dan penggunaan logika fuzzy untuk control

Nilai

Logika fuzzy berasal dari beberapa pengamatan, yaitu:

- Pengetahuan yang dimiliki manusia tentang situasi umumnya tidak sempurna,
    - bisa tidak pasti (ia meragukan validitasnya),
    - atau tidak tepat.
- Manusia sering memecahkan masalah yang kompleks dengan data perkiraan: akurasi data sering tidak berguna; misalnya, untuk memilih apartemen dia mungkin memperhitungkan permukaan area, kedekatan pertokoan, jarak dari tempat kerja dan sewa tanpa, bagaimanapun, membutuhkan nilai yang sangat tepat untuk setiap informasi.
- Dalam industri dan teknologi, operator sering memecahkan masalah yang kompleks dalam waktu yang relative cara sederhana tanpa perlu model sistem. Demikian juga, sudah menjadi rahasia umum bahwa model matematika tidak diperlukan untuk mengemudi mobil, namun mobil adalah sistem yang sangat kompleks.
- Semakin kompleks suatu sistem, semakin sulit itu adalah untuk membuat pernyataan yang tepat tentang perilakunya.

Berikut ini secara alami disimpulkan dari ini pengamatan:

- daripada memodelkan sistem, seringkali lebih berguna untuk memodelkan perilaku manusia operator yang digunakan untuk mengontrol sistem;
- daripada menggunakan persamaan, operasi dapat dijelaskan secara kualitatif dengan terjemahan kuantitatif.

Digunakan untuk tujuan kontrol

Logika fuzzy dikenal dengan kontrol otomatis insinyur untuk aplikasinya dalam kontrol proses dan pemantauan, maka biasa disebut sebagai "kontrol kabur". Sama seperti konvensional kontroler, kontroler fuzzy tergabung dalam loop kontrol dan menghitung kontrol menjadi diterapkan pada proses menurut satu atau lebih setpoint dan satu atau lebih pengukuran yang diambil pada proses.

Basis aturan fuzzy menguntungkan dalam kontrol karena mereka mengizinkan:

1. pertimbangan keahlian kualitatif yang ada,
2. pertimbangan variabel yang efeknya akan sulit untuk dimodelkan dengan cara tradisional, tetapi dikenal secara kualitatif,
3. peningkatan pengontrol konvensional operasi oleh:
   penyetelan sendiri keuntungan pengontrol off line atau on line,
   modifikasi outputnya (feed forward) sesuai dengan peristiwa yang tidak dapat dianggap akun menggunakan teknik konvensional.

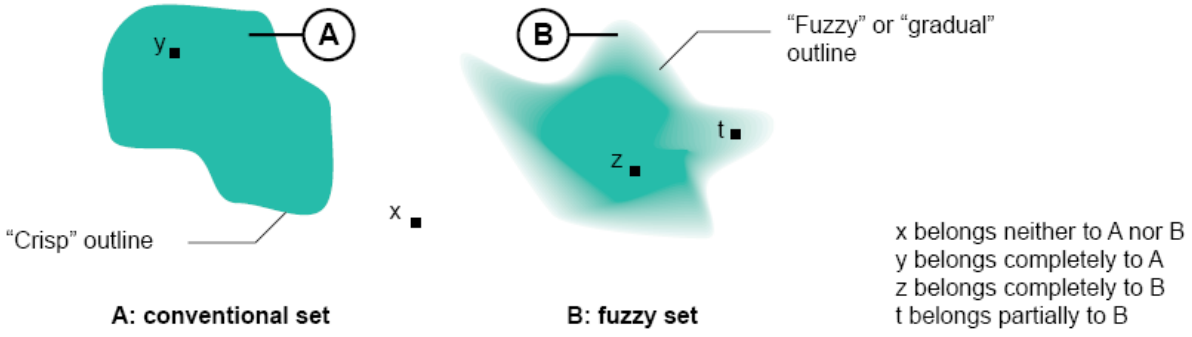
Menggunakan pengetahuan untuk keuntungan terbaik

Kondisi vital untuk penggunaan aturan fuzzy adalah adanya keahlian dan pengetahuan manusia. Basis aturan fuzzy tidak dapat memberikan solusi ketika tidak ada yang tahu bagaimana sistem beroperasi atau orang tidak dapat mengontrolnya secara manual. Ketika pengetahuan seperti itu ada dan dapat terjadi ditranskripsikan dalam bentuk aturan fuzzy, logika fuzzy menyederhanakan implementasinya, dan operasinya adalah kemudian mudah dipahami oleh pengguna. Logika fuzzy juga memungkinkan keuntungan maksimal menjadi berasal dari pengetahuan praktis, sering dicari untuk mencegah hilangnya pengetahuan atau untuk berbagi pengetahuan ini dengan orang lain di perusahaan. Saat mengumpulkan keahlian, kelalaian yang tidak disadari informasi, kesulitan untuk menjelaskan dan ketakutan untuk mengungkapkan pengetahuan adalah hambatan yang sering ditemui. Tahap ini harus karena itu bersiaplah dan lakukan dengan hati-hati, dengan mempertimbangkan faktor manusia. Jika keahlian manusia ada, maka aturan fuzzy bias digunakan, terutama ketika pengetahuan system dinodai oleh ketidaksempurnaan, ketika sistemnya kompleks dan sulit untuk dimodelkan dan ketika metode yang digunakan membutuhkan pandangan global dari beberapa aspek-aspeknya. Aturan fuzzy tidak menggantikan metode kontrol otomatis konvensional, bukan mereka menyelesaikan metode ini.

7.2 Teori Logika Fuzzy

7.2.1 Pengertian keanggotaan parsial

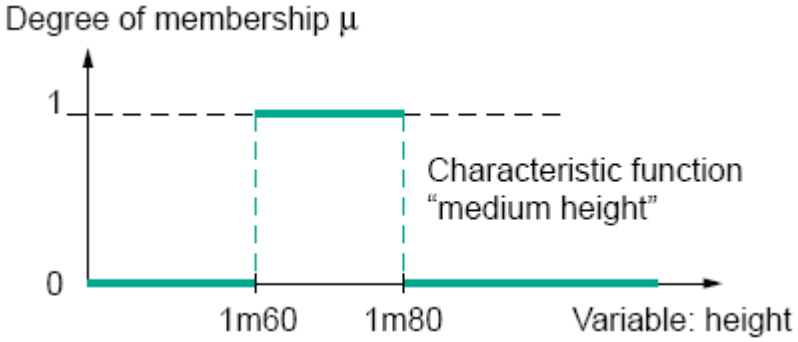
Dalam teori himpunan, suatu elemen termasuk atau bukan milik himpunan. Pengertian himpunan adalah digunakan dalam banyak teori matematika. Ini gagasan penting, bagaimanapun, tidak memperhitungkan situasi akun yang sederhana dan umum. Berbicara tentang buah-buahan, mudah untuk didefinisikan himpunan apel. Namun, lebih sulit untuk didefinisikanset apel matang. Kami memahami bahwa apel matang secara progresif ... gagasan tentang matang apel dengan demikian adalah yang bertahap.
Gagasan tentang himpunan kabur dibuat untuk mempertimbangkan situasi semacam ini. Itu teori himpunan fuzzy didasarkan pada gagasan keanggotaan parsial: setiap elemen milik sebagian atau secara bertahap ke himpunan fuzzy yang memiliki telah didefinisikan. Garis besar setiap himpunan fuzzy (lihat gbr 7.1 tidak "garing", tetapi "kabur" atau "bertahap"



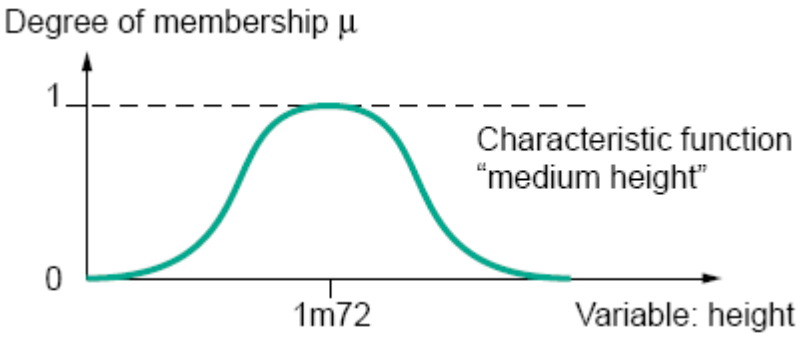
Gambar 7.1 : perbandingan himpunan konvensional dan himpunan fuzzy.

7.2.2 Fungsi keanggotaan Logika Fuzzy

Himpunan fuzzy didefinisikan oleh "keanggotaannya" fungsi" yang sesuai dengan gagasan a "fungsi karakteristik" dalam logika klasik. Mari kita asumsikan bahwa kita ingin mendefinisikan himpunan orang-orang dengan "tinggi sedang". Dalam logika klasik, kita akan setuju misalnya bahwa orang-orang menengah tinggi adalah antara 1,60 m dan 1,80 m tinggi. Fungsi karakteristik himpunan (lihat gambar 7.2) memberikan "0" untuk ketinggian di luar kisaran [1,60 m; 1,80 m] dan "1" untuk ketinggian di dalamnya jangkauan. Himpunan kabur dari orang-orang "sedang" tinggi" akan ditentukan oleh "keanggotaan fungsi" yang berbeda dari karakteristik berfungsi karena dapat mengasumsikan nilai apa pun dalam rentang [0;1]. Setiap ketinggian yang mungkin akan diberi "derajat" keanggotaan" ke himpunan fuzzy "medium" ketinggian" (lihat gambar 7.3) antara 0 dan 1.

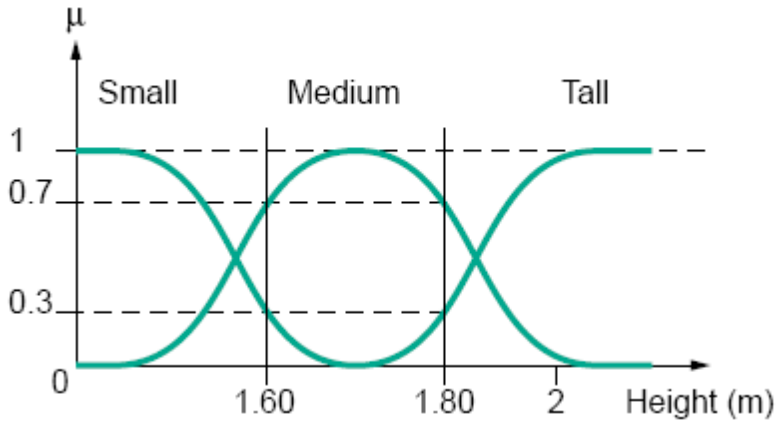


Gambar 7.2 : fungsi karakteristik.



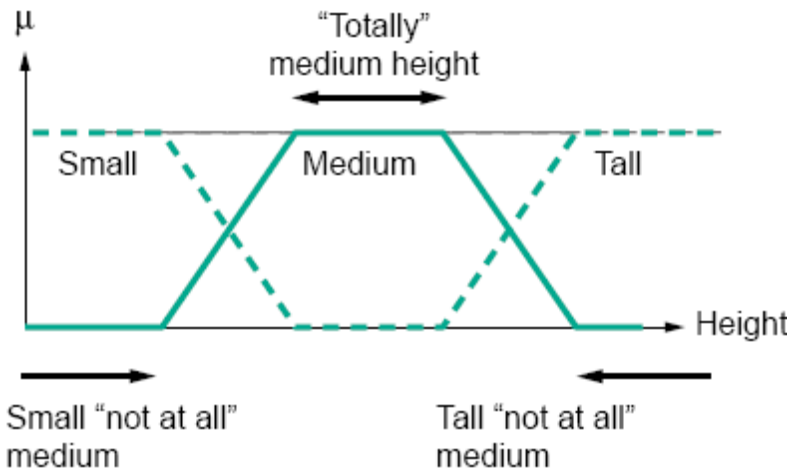
Gambar 7.3 : fungsi keanggotaan.

Sejumlah himpunan fuzzy dapat didefinisikan pada variabel yang sama, misalnya himpunan “kecil” tinggi", "tinggi sedang" dan "tinggi", masing-masing gagasan yang dijelaskan oleh fungsi keanggotaan (lihat gambar 7.4).



Gambar 7.4 : fungsi keanggotaan, variabel dan suku kebahasaan.

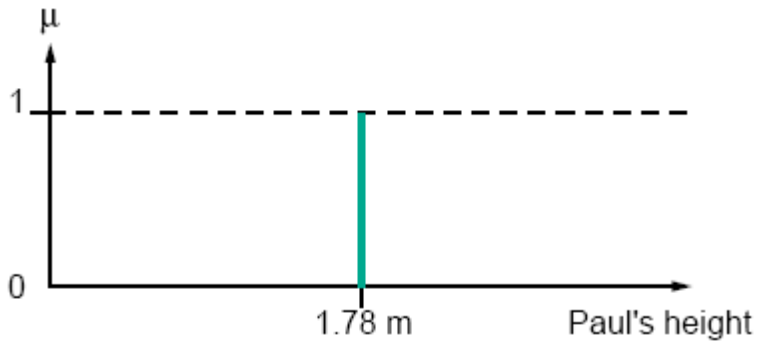
Contoh ini menunjukkan bertahap yang memungkinkan logika fuzzy yang akan diperkenalkan. Seseorang dengan tinggi 1,80 m milik set "tinggi" dengan derajat 0,3, dan ke set "tinggi sedang" dengan derajat 0,7. Dalam logika klasik, perubahan dari rata-rata menjadi tinggi akan tiba-tiba. Seseorang dengan ketinggian 1,80 m kemudian akan menjadi tinggi sedang, sedangkan orang 1,81 m akan tinggi, pernyataan yang mengejutkan intuisi. Variabel (misalnya: tinggi) sebagai serta istilah (misalnya: sedang, tinggi) didefinisikan oleh fungsi keanggotaan, diketahui sebagai variabel linguistik dan istilah linguistik masing-masing. Seperti yang akan kita lihat lebih lanjut, baik linguistik variabel dan istilah dapat digunakan secara langsung dalam aturan. Fungsi keanggotaan dapat mengambil bentuk apapun. Namun mereka sering didefinisikan oleh straight segmen dan dikatakan "linear potong-bijaksana" (lihat gambar 7.5). Fungsi keanggotaan “Piece-wise linear” adalah sering digunakan sebagai:



Gambar 7.5 : fungsi keanggotaan linier piece-wise.

- sederhana,
- berisi poin yang memungkinkan definisi area

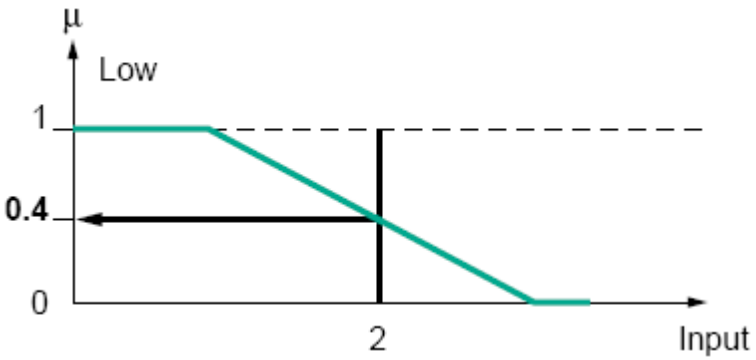
di mana gagasan itu benar dan area di mana itu salah, sehingga menyederhanakan pengumpulan keahlian. Kami telah memilih untuk menggunakan fungsi keanggotaan dari semacam ini di sisa dokumen ini. Dalam beberapa kasus, fungsi keanggotaan mungkin: sama dengan 1 untuk satu nilai variabel, dan sama dengan 0 di tempat lain. Mereka kemudian dikenal sebagai "fungsi keanggotaan tunggal". kabur
singleton (lihat gambar 6) didefinisikan pada variabel nyata (tinggi) adalah ekspresi dalam bidang fuzzy dari a nilai spesifik (tinggi Paulus) dari variabel ini (Lihat Lampiran).



Gambar 7.6 : fungsi keanggotaan tunggal.

Fuzzifikasi - Derajat keanggotaan

Fuzzifikasi memungkinkan nilai nyata menjadi diubah menjadi kabur. Ini terdiri dari menentukan derajat keanggotaan suatu nilai (diukur dengan contoh) ke himpunan kabur. Misalnya (lihat gambar 7.7), jika nilai saat ini dari variabel "input" adalah 2, the derajat keanggotaan ke "input rendah" fungsi keanggotaan sama dengan 0,4 yang merupakan hasil dari fuzzifikasi. Kami juga dapat mengatakan bahwa proposal "masukan rendah" adalah benar pada 0,4. Kami kemudian berbicara tentang tingkat kebenaran dari usul. Derajat keanggotaan dan derajat kebenaran karena itu adalah gagasan yang serupa.



Gambar 7.7 : fuzzifikasi.

7.2.3 Operator logika fuzzy

Operator ini digunakan untuk menulis logika kombinasi antara gagasan fuzzy, yaitu untuk melakukan perhitungan pada derajat kebenaran. Hanya seperti untuk logika klasik, AND, OR dan NOT operator dapat ditentukan.

Contoh: Apartemen Menarik = Sewa Yang Wajar DAN Luas Permukaan Yang Cukup.

Pilihan operator

Operator ini memiliki banyak varian (lihat lampiran). Namun yang paling umum adalah

Operator “Zadeh” dijelaskan di bawah ini. Derajat kebenaran proposal A adalah dicatat m(A).

Persimpangan Operator logika yang sesuai dengan perpotongan himpunan adalah AND. Derajat kebenaran dari proposal "A DAN B" adalah nilai minimum derajat kebenaran A dan B :

m(A DAN B) = MIN(m(A),m(B))

Sebagai contoh:

"Suhu Rendah" benar pada 0,7

"Tekanan Rendah" benar pada 0,5

"Suhu Rendah DAN Tekanan Rendah" adalah

oleh karena itu benar pada 0,5 = MIN(0,7; 0,5).

NB: fuzzy AND ini kompatibel dengan klasik logika 0 dan 1, yeld 0.

Union

Operator logika yang sesuai dengan gabungan himpunan adalah ATAU. Tingkat kebenaran proposal "A OR B" adalah nilai maksimum derajat kebenaran A dan B:

m(A ATAU B) = MAX(m(A),m(B))

Sebagai contoh:

"Suhu Rendah" benar pada 0,7

"Tekanan Rendah" benar pada 0,5

"Suhu Rendah ATAU Tekanan Rendah" adalah oleh karena itu benar pada 0,7.

NB: fuzzy OR ini kompatibel dengan klasik

logika: 0 ATAU 1 menghasilkan 1.

Complement

Operator logika yang sesuai dengan

komplemen suatu himpunan adalah negasi.

m(BUKAN A) = 1 - m(A)

Sebagai contoh:

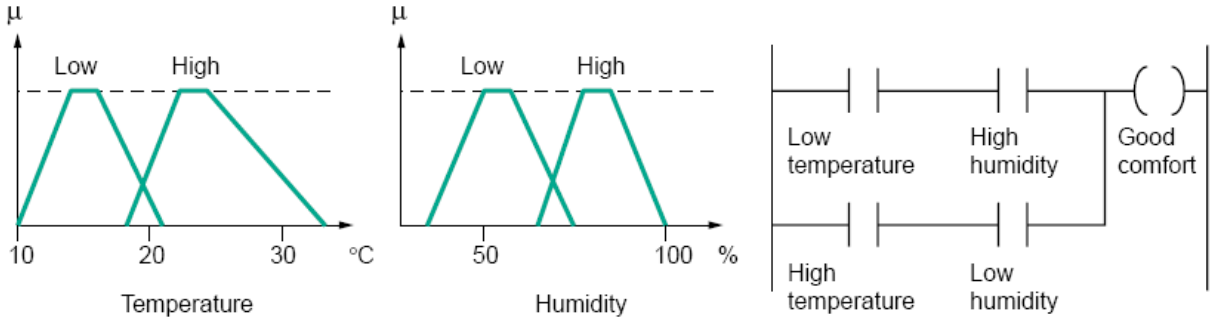
"Suhu Rendah" benar pada 0,7

"TIDAK Suhu Rendah" yang biasanya kita lakukan tulis sebagai "Suhu TIDAK Rendah" karena itu benar pada 0,3.

NB: operator negasi kompatibel dengan logika klasik: NOT(0) menghasilkan 1 dan NOT(1) menghasilkan 0.

Fuzzy ladder

Bahasa tangga atau bahasa kontak adalah biasa digunakan oleh insinyur kontrol otomatis untuk menulis kombinasi logika, karena memungkinkan mereka representasi grafis. Schneider memiliki memperkenalkan penggunaan representasi tangga untuk menjelaskan kombinasi logika fuzzy. Di bawah ini adalah contoh yang berhubungan dengan kenyamanan udara sekitar: panas, udara lembab tidak nyaman (berlebihan keringat); demikian juga bernapas sulit di udara yang dingin dan terlalu kering. Yang paling nyaman situasi termal adalah situasi di mana udara panas dan kering, atau dingin dan lembab. Ini bisa jadi ditranskripsikan oleh tangga kabur pada gambar 8 sesuai dengan kombinasi berikut: Kenyamanan yang baik = (Suhu Rendah DAN Tinggi Kelembaban) ATAU (Suhu Tinggi DAN Rendah Kelembaban). Ini mewakili definisi yang mungkin dari sensasi kenyamanan yang dirasakan oleh seseorang dalam keadaan termal lingkungan di mana udara tidak bergerak.



Gambar 7.8 : **Fuzzy ladder**

Klasifikasi **Fuzzy**

Klasifikasi biasanya terdiri dari dua langkah:

- persiapan: menentukan kelas yang akan dipertimbangkan,
- on line: menugaskan elemen ke kelas.

Pengertian kelas dan himpunan adalah identic secara teoretis.

Ada tiga jenis metode penugasan: sesuai dengan hasil yang dihasilkan:

- boolean: elemen termasuk atau tidak milik kelas,
- probabilistik: elemen memiliki probabilitas milik kelas boolean, seperti misalnya probabilitas bahwa seorang pasien menderita campak diberikan gejala yang ditunjukkannya (diagnosis),
- bertahap: elemen memiliki derajat keanggotaan ke set; misalnya selada.

milik tingkat yang berbeda-beda ke kelas "segar" selada".

Metode klasifikasi, apakah mereka menghasilkan hasil bertahap, boolean atau probabilistik, dapat berupa dikembangkan dari:

- percobaan (kasus "tangga kabur" disebutkan di atas),
- contoh yang digunakan untuk tujuan pembelajaran (misalnya untuk pengklasifikasi jaringan neuron),
- pengetahuan matematika atau fisik dari a masalah (misalnya, kenyamanan termal situasi dapat dievaluasi dari keseimbangan termal persamaan).

Metode klasifikasi bertahap (atau kabur) dapat digunakan dalam loop kontrol. Ini adalah kasus dari contoh masakan industri untuk biskuit dijelaskan kemudian.

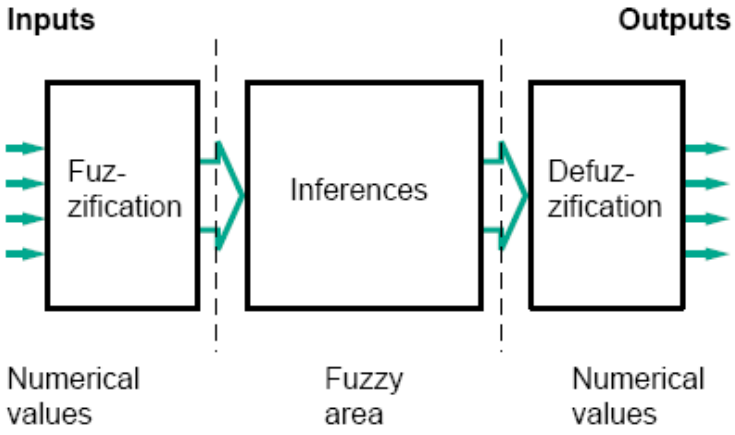
7.2.4 Aturan Fuzzy

Logika kabur dan kecerdasan buatan Tujuan dari basis aturan fuzzy adalah untuk memformalkan dan menerapkan metode manusia pemikiran. Karena itu dapat diklasifikasikan di lapangan dari kecerdasan buatan. Alat yang paling umum digunakan dalam logika fuzzy aplikasi adalah basis aturan fuzzy. Aturan

kabur dasar terbuat dari aturan yang biasanya digunakan dalam paralel tetapi juga dapat digabungkan menjadi beberapa aplikasi.
Aturan adalah dari jenis:
JIKA "predikat" MAKA "kesimpulan".
Misalnya: JIKA "suhu tinggi dan tinggi"
tekanan" KEMUDIAN "ventilasi kuat dan lebar katup terbuka".
Basis aturan fuzzy, seperti ahli konvensional sistem, mengandalkan basis pengetahuan yang berasal dari keahlian manusia. Namun demikian, ada mayor perbedaan karakteristik dan pengolahan pengetahuan ini (lihat gambar 7.9). Aturan fuzzy terdiri dari tiga bagian yang tidak berfungsi: diringkas dalam gambar 7. 10.

Fuzzy rule base	**Conventional rule base (expert system)**
Few rules	Many rules
Gradual processing	Boolean processing
Concatenation possible but scarcely used	Concatenated rules A OR B ⇒ C, C ⇒ D, D AND A ⇒ E
Rules processed in parallel	Rules used one by one, sequentially
Interpolation between rules that may contradict one another	No interpolation, no contradiction

Gambar 7.9 : basis aturan fuzzy dan basis aturan konvensional.



Gambar 7.10 : pemrosesan fuzzy.

Predicate
Predikat (juga dikenal sebagai premis atau kondisi) adalah kombinasi proposal oleh AND, ATAU, BUKAN operator. "Suhu tinggi" dan "tekanan tinggi" proposal dalam contoh sebelumnya digabungkan oleh operator AND untuk membentuk predikat dari aturan.

Inference
Mekanisme inferensi yang paling umum digunakan adalah "Mamdani". Ini mewakili penyederhanaan mekanisme yang lebih umum berdasarkan "implikasi kabur" dan "modus ponens umum". Konsep-konsep ini dijelaskan dalam lampiran. Hanya Basis aturan "Mamdani" digunakan di bawah ini.

Kesimpulan
Kesimpulan dari aturan fuzzy adalah kombinasi dari proposal yang ditautkan oleh operator AND. Dalam contoh sebelumnya, "ventilasi kuat" dan "lebar" katup terbuka" adalah kesimpulan dari aturan. Klausa "ATAU" tidak digunakan dalam kesimpulan karena akan memperkenalkan ketidakpastian ke dalam pengetahuan (keahlian tidak akan membuatnya mungkin untuk menentukan keputusan mana yang harus dibuat). Ketidakpastian ini tidak diperhitungkan oleh mekanisme inferensi Mamdani yang hanya mengelola

ketidaktepatan. Oleh karena itu Aturan fuzzy "Mamdani" secara teori tidak cocok untuk diagnosis jenis "medis" yang kesimpulannya tidak pasti. teori dari kemungkinan, ditemukan oleh Lotfi Zadeh, menawarkan metodologi yang tepat dalam kasus tersebut. Demikian juga, negasi tidak digunakan dalam kesimpulan untuk aturan Mamdani. Ini karena jika aturan itu memiliki kesimpulan “Maka ventilasi tidak rata-rata", tidak mungkin untuk mengatakan apakah ini berarti "ventilasi lemah" atau "kuat" ventilasi". Ini akan menjadi kasus lain dari ketakpastian.

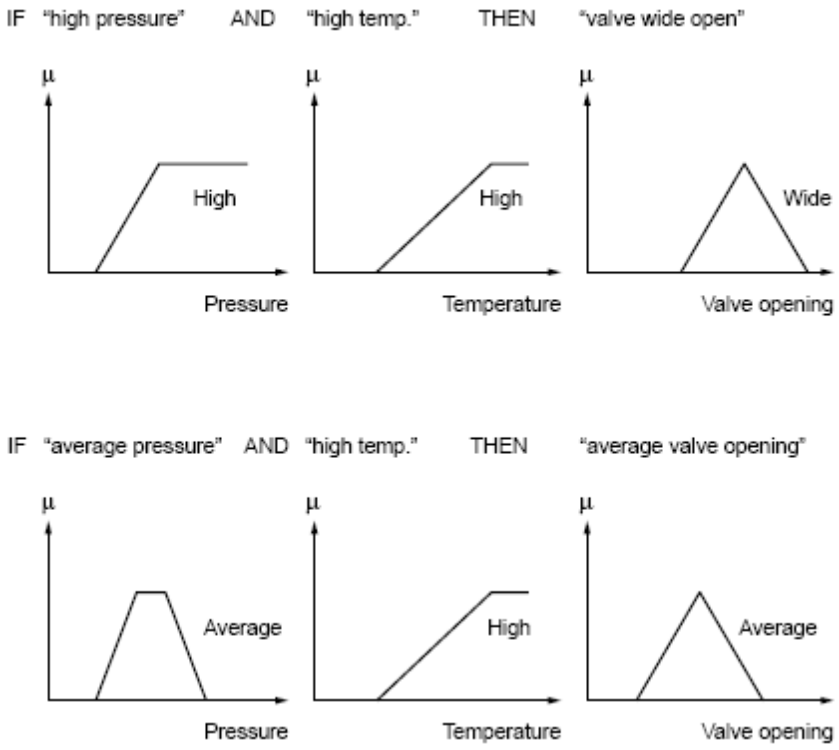
Mekanisme inferensi Mamdani

- Prinsip

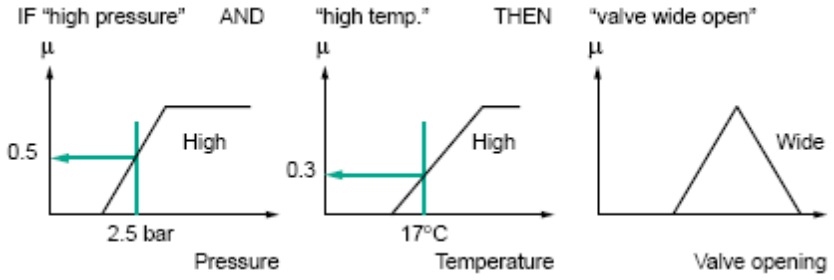
Oleh karena itu, basis aturan fuzzy Mamdani mengandung aturan linguistik menggunakan fungsi keanggotaan untuk jelaskan konsep yang digunakan (lihat gambar 7.11). Mekanisme inferensi terdiri dari langkah-langkah berikut:

- Fuzzifikasi

Fuzzifikasi terdiri dari mengevaluasi fungsi keanggotaan yang digunakan dalam predikat aturan, sebagai diilustrasikan pada gambar 7.12 :



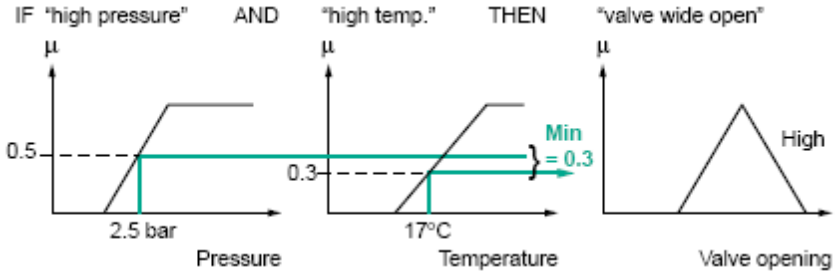
Gambar 7.11 : implikasi.



Gambar 7.11 : fuzzifikasi.

Tingkat aktivasi

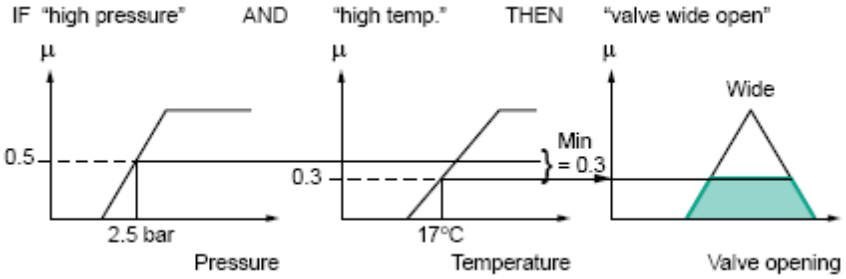
Derajat aktivasi suatu aturan adalah evaluasi predikat setiap aturan dengan logika kombinasi predikat proposal, seperti yang ditunjukkan pada gambar 7.13 . “DAN” dilakukan dengan mewujudkan minimum antara derajat kebenaran proposal.



Gambar 7.12 : aktivasi.

Implikasi

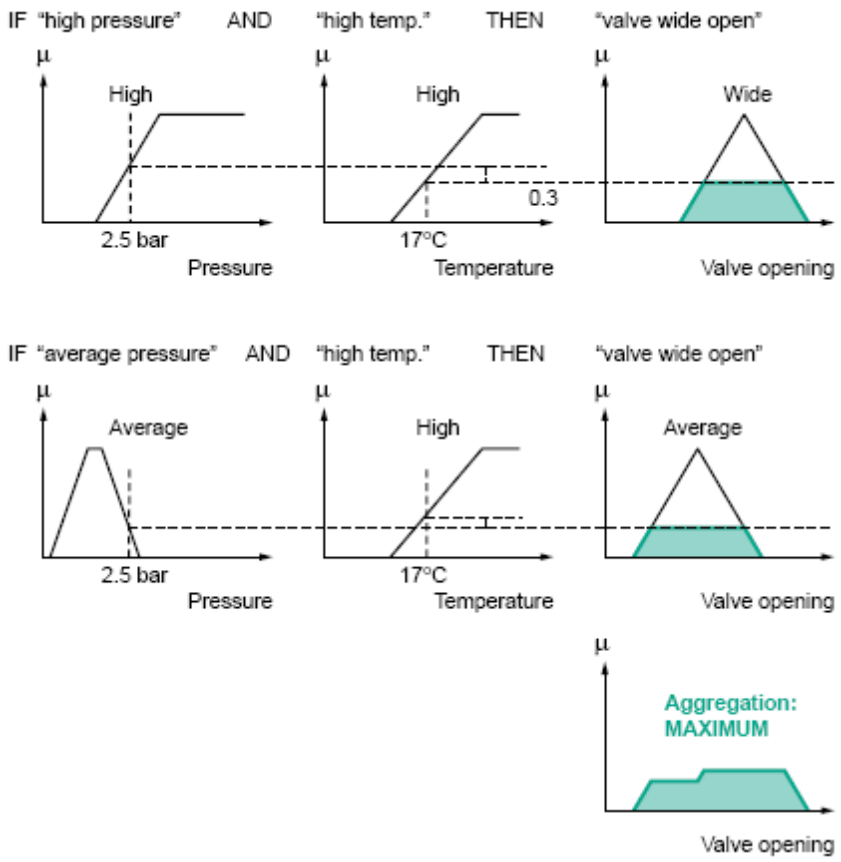
Derajat aktivasi aturan digunakan untuk tentukan kesimpulan dari aturan: ini operasi disebut implikasi. Ada beberapa operator implikasi (lihat lampiran), tetapi yang paling umum adalah operator "minimum". Himpunan fuzzy kesimpulan dibangun dengan merealisasikan minimum antara tingkat aktivasi dan fungsi keanggotaan, semacam "kliping" dari fungsi keanggotaan kesimpulan (lihat gambar 7.14).



Gambar 7.13 : implikasi.

agregasi

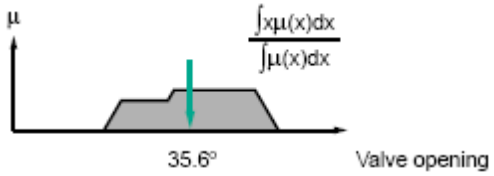
Himpunan fuzzy global keluaran dibangun oleh agregasi himpunan fuzzy yang diperoleh masing-masing aturan tentang keluaran ini. Contoh di bawah ini menunjukkan kasus ketika dua aturan bertindak atas keluaran. Aturan dianggap terkait oleh logika "ATAU", dan karena itu kami menghitung nilai maksimum antara yang dihasilkan fungsi keanggotaan untuk setiap aturan (lihat gambar 7.15).



Gambar 7.14 : agregasi aturan.

Defuzzifikasi

Pada akhir inferensi, himpunan fuzzy keluarannya adalah ditentukan, tetapi tidak dapat langsung digunakan untuk memberikan operator dengan informasi yang tepat atau mengontrol sebuah aktuator. Kita harus pindah dari "dunia kabur" ke "dunia nyata": ini dikenal sebagai defuzzifikasi. Sejumlah metode dapat digunakan, yang paling umum di antaranya adalah perhitungan "pusat" gravitasi" dari himpunan fuzzy (lihat gambar 7.16).



Gambar 7.15 : defuzzifikasi oleh pusat gravitasi.

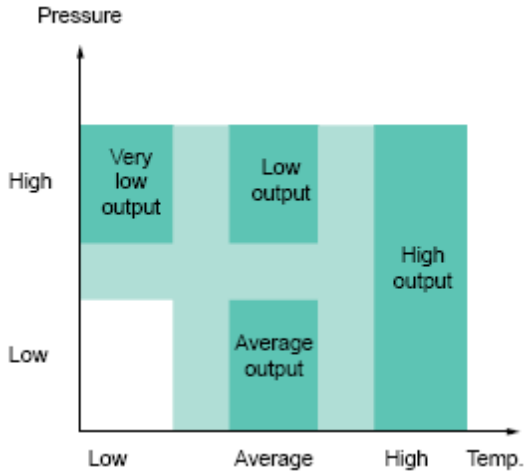
Aturan "Gratis" dan "mampu"

Basis aturan fuzzy, dalam kasus umum mereka, gunakan fungsi keanggotaan pada variabel sistem, dan aturan yang dapat ditulis secara tekstual. Setiap aturan menggunakan input dan outputnya sendiri, seperti yang ditunjukkan oleh contoh di bawah ini:

R1: JIKA "suhu tinggi"
KEMUDIAN "keluaran tinggi"
R2: JIKA "suhu rata-rata"
DAN "tekanan rendah"

LALU "keluaran rata-rata"
R3: JIKA “suhu rata-rata”
DAN “tekanan tinggi”
KEMUDIAN "keluaran rendah"
R4: JIKA “suhu rendah”
DAN “tekanan tinggi”
KEMUDIAN "keluaran sangat rendah"

Dalam bentuk diagram, "bidang tindakan" dari aturan dan tumpang tindihnya dapat direpresentasikan dalam tabel pada gambar 7.17.



Gambar 7.16 : implikasi direpresentasikan dalam sebuah tabel.

Kita dapat melakukan pengamatan berikut:

- tidak semua ruang harus tertutup: the kombinasi "suhu rendah dan tekanan rendah" tidak diperhitungkan dalam kasus ini. Itu penjelasannya adalah misalnya kombinasi ini tidak mungkin secara fisik untuk mesin ini atau tidak menarik bagi kami. Yang terbaik adalah memverifikasinya seperti ini mungkin kelalaian;
- aturan pertama hanya memperhitungkan suhu akun: situasi ini normal karena mencerminkan keahlian yang ada. Namun, banyak aplikasi mendefinisikan aturan "tabel". Dalam konteks ini, ruang itu "digrid" dan masing-masing "kotak" di kotak diberi aturan. Ini pendekatan memiliki keuntungan menjadi sistematis, tetapi:
    - itu tidak selalu memungkinkan ekspresi sederhana (dalam jumlah minimum aturan) dari yang ada keahlian,
    - hanya dapat diterapkan untuk dua atau tiga input, sedangkan basis aturan "bebas" dapat dibangun dengan a sejumlah besar variabel.
- Perilaku basis aturan fuzzy adalah statis dan non-linier sehubungan dengan inputnya. - Basis aturan fuzzy tidak sendiri dinamis, meskipun mereka sering digunakan sebagai variabel input mengekspresikan dinamika sistem (turunan, integral, dll.) atau waktu.
- Keuntungan utama dari "PID fuzzy" controller, sering disajikan sebagai pengajaran Contoh untuk memberikan gambaran tentang logika fuzzy, adalah dengan membuat PID non-linear, yang jarang membenarkan penggunaannya di tempat PID konvensional. Apalagi itu akan sulit untuk menggabungkan yang sudah ada keahlian dalam hal ini.
-

7.3 Contoh aplikasi pengajaran

7.3.1 Pendahuluan

Sebagian besar pencapaian logika fuzzy memerlukan pendahuluan
pengetahuan khusus tentang area aplikasi. Agar mudah dipahami oleh pembaca, contoh berikut dibuat berdasarkan aplikasi fiktif dan dirancang untuk menggambarkan prosedur pembuatan basis aturan fuzzy.

7.3.2 Presentasi contoh

Contoh tersebut berkaitan dengan proses pencucian selada untuk produksi selada kemasan di konter "produk segar" supermarket.
Selada dipotong, dicuci dan dikemas. Tujuan pencucian adalah untuk menghilangkan tanah dari selada serta mikroorganisme yang dapat berkembang biak selama masa simpan produk. Pabrikan ingin mengotomatiskan proses pencucian.
Mencuci adalah proses yang berkesinambungan. Daun selada ditempatkan di "drum" yang bergerak melalui "terowongan" yang dilengkapi dengan nozel penyemprotan
air yang diklorinasi. Air menghilangkan tanah, sedangkan klorin membunuh mikroorganisme (lihat gambar 7.18).
Prioritas berikut dirumuskan oleh departemen pemasaran dan diurutkan berdasarkan kepentingannya:

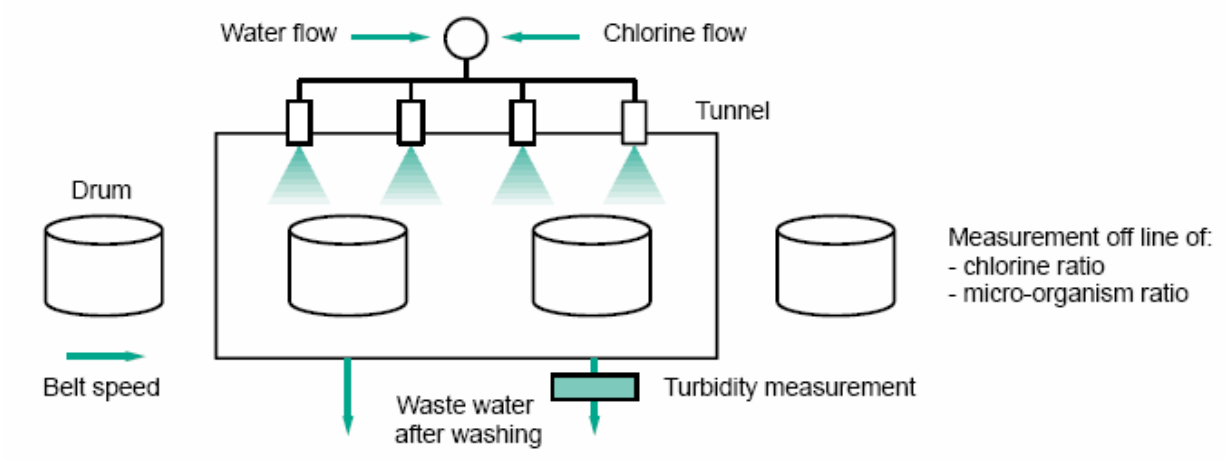
- Sehubungan dengan pelanggan
- kualitas jaminan
    - Selada "sangat bersih" (penampilan)
    - Tidak ada rasa klorin.
- Jaminan keamanan
    - Tingkat mikroorganisme yang dapat diterima
- Sehubungan dengan profitabilitas
- Memaksimalkan produksi
- Hemat air
- Hemat klorin.

Operator secara manual mengendalikan proses biasanya memeriksa air kotor di akhir pencucian terowongan. Jika airnya jernih, mereka menyimpulkan berdasarkan pengalaman bahwa selada akan memiliki penampilan "bersih". Keputusan demikian dibuat untuk memasang sensor optik "kekeruhan" yang dirancang untuk tentukan derajat transparansi dari air.
Selain itu, operator menggunakan laporan satu jam sekali berdasarkan analisis yang dilakukan di pabrik yang memberikan rasio mikroorganisme dan residu klorin ditemukan dalam selada yang dicuci dan dicuci sebelumnya di akhir baris.
Oleh karena itu tujuannya adalah untuk menggunakan yang di atas informasi untuk meningkatkan pengendalian:

- kecepatan sabuk konveyor selada (untuk meningkatkan hasil produksi),
- jumlah klorin yang disemprotkan,
- jumlah air yang disemprotkan.

Batas yang dikenakan:

- pada kecepatan ban berjalan, dengan mekanisme,
- c pada aliran air untuk mencegah kerusakan selada daun-daun.



Gambar 7.17 : Proses pencucian selada.

7.3.3. Variabel dan istilah linguistik

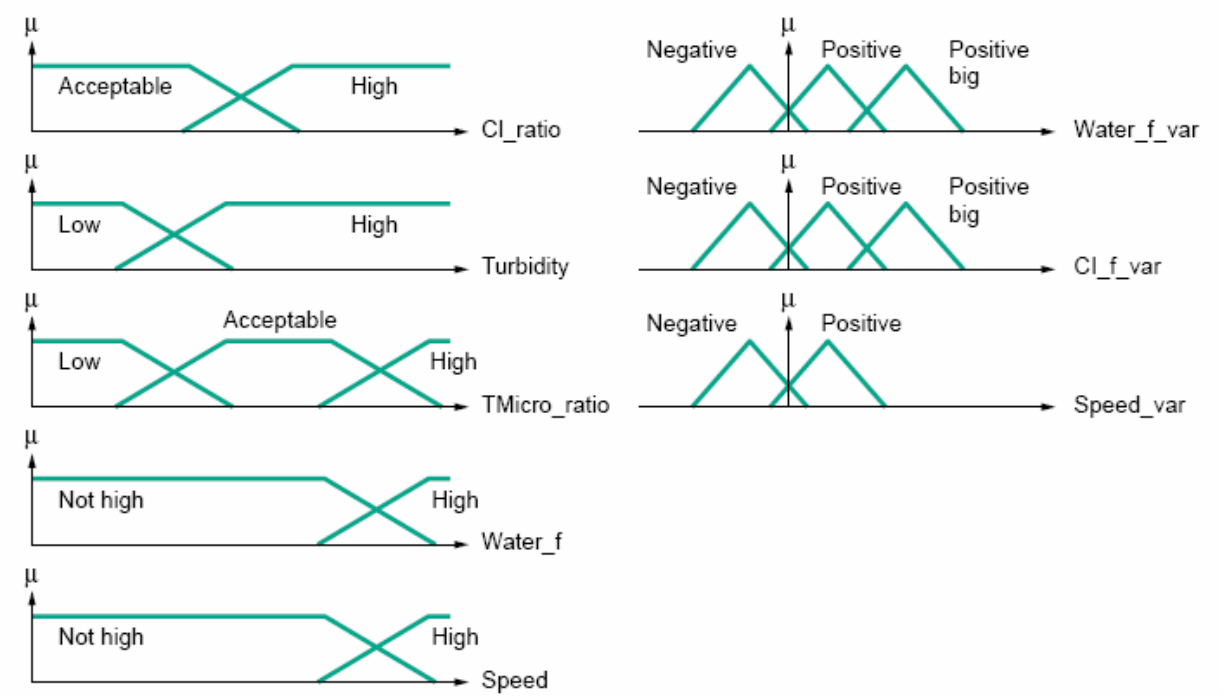
Oleh karena itu, variabel berikut akan menjadi terpilih:
Masukan :

- rasio mikroorganisme: Rasio_mikro
- rasio sisa klorin: Cl_ratio
- kekeruhan air: Kekeruhan
- kecepatan sabuk konveyor: Kecepatan
- aliran air: Water_f

keluaran:

- modifikasi aliran air: Water_f_var
- modifikasi aliran klorin: Cl_f_var
- modifikasi kecepatan: Kecepatan-var

Sesi dengan operator berpengalaman, spesialis mikrobiologi, dan "pencicip" selada menghasilkan fungsi keanggotaan berikut: (lihat gambar 7.19):



Gambar 7.18 : fungsi keanggotaan linier piece-wise.

7.3.4. Aturan dan keluaran

Menulis aturan fuzzy

Pertemuan dengan operator memungkinkan tujuh aturan di bawah ini untuk ditentukan, masing-masing sesuai dengan kasus tertentu:

- Selada dicuci dengan buruk
  JIKA Kekeruhan Tinggi DAN Water_f TIDAK Tinggi
  MAKA Water_f_var IS Positif besar.
- Selada dicuci dengan buruk tetapi sabuk konveyor tinggi kecepatan
  JIKA Kekeruhan Tinggi DAN Water_f Tinggi MAKA
  Speed_var IS Negatif.
- Terlalu banyak mikroorganisme
  JIKA Rasio_mikro Tinggi MAKA Cl_f_var Positif besar.
- Semuanya baik-baik saja dan produksi bias ditingkatkan
  JIKA Kekeruhan Rendah dan Micro_ratio TIDAK Tinggi
  DAN Kecepatan TIDAK Tinggi dan CL_ratio ADALAH
  Dapat diterima DAN Water_f TIDAK TINGGI LALU
  Speed_var Positif DAN Cl_f_var Positif
  DAN Water_f_var Positif.
- Selada rasanya klorin, tapi tidak ada mikroorganisme
  JIKA Cl_ratio TINGGI DAN Micro_ratio TIDAK TINGGI
  MAKA Cl_f_var ADALAH Negatif.
- Semuanya baik-baik saja dan produksi maksimal:
  hemat air.
  JIKA Kecepatan Tinggi DAN Cl_ratio Dapat Diterima
  DAN Kekeruhannya Rendah MAKA Water_f_var IS Negatif.

- Tidak ada mikroorganisme: hemat klorin
  Jika Micro_ratio Rendah MAKA Cl_f_var Negatif.

Defuzzifikasi

Sejauh tujuannya adalah perilaku progresif dari basis aturan dalam semua kasus dan interpolasi antara aturan, pusat gravitasi dipilih sebagai operator defuzzifikasi.

Studi Kasus

Untuk menjelaskan penggunaan Fuzzy Logic Toolbox tersebut, perhatikan contoh studi kasus berikut. Romi ingin mengajak Juli dinner di suatu restoran. Romi ingin membagi kebahagiaan pada malam ini kepada pelayan restoran dengan memberikan uang tip. Dia akan memberikan uang tip sebesar **5-25%** dari total belanjanya. Besarnya uang tip diberikan berdasarkan tingkat **PELAYANAN** dan kualitas **MAKANAN** yang dihidangkan.

Bantulah Romi untuk memutuskan besarnya uang tip, jika setelah menikmati hidangan dan fasilitas PELAYANAN Romi memberi nilai:

PELAYANAN = 7

MAKANAN = 8

Penilaian PELAYANAN dan MAKANAN berada pada rentang nilai 0-10 dan kisaran uang tip adalah 5 - 25%.

Adapun aturan pemberian tip yg ditetapkan oleh Romi adalah sebagai berikut:

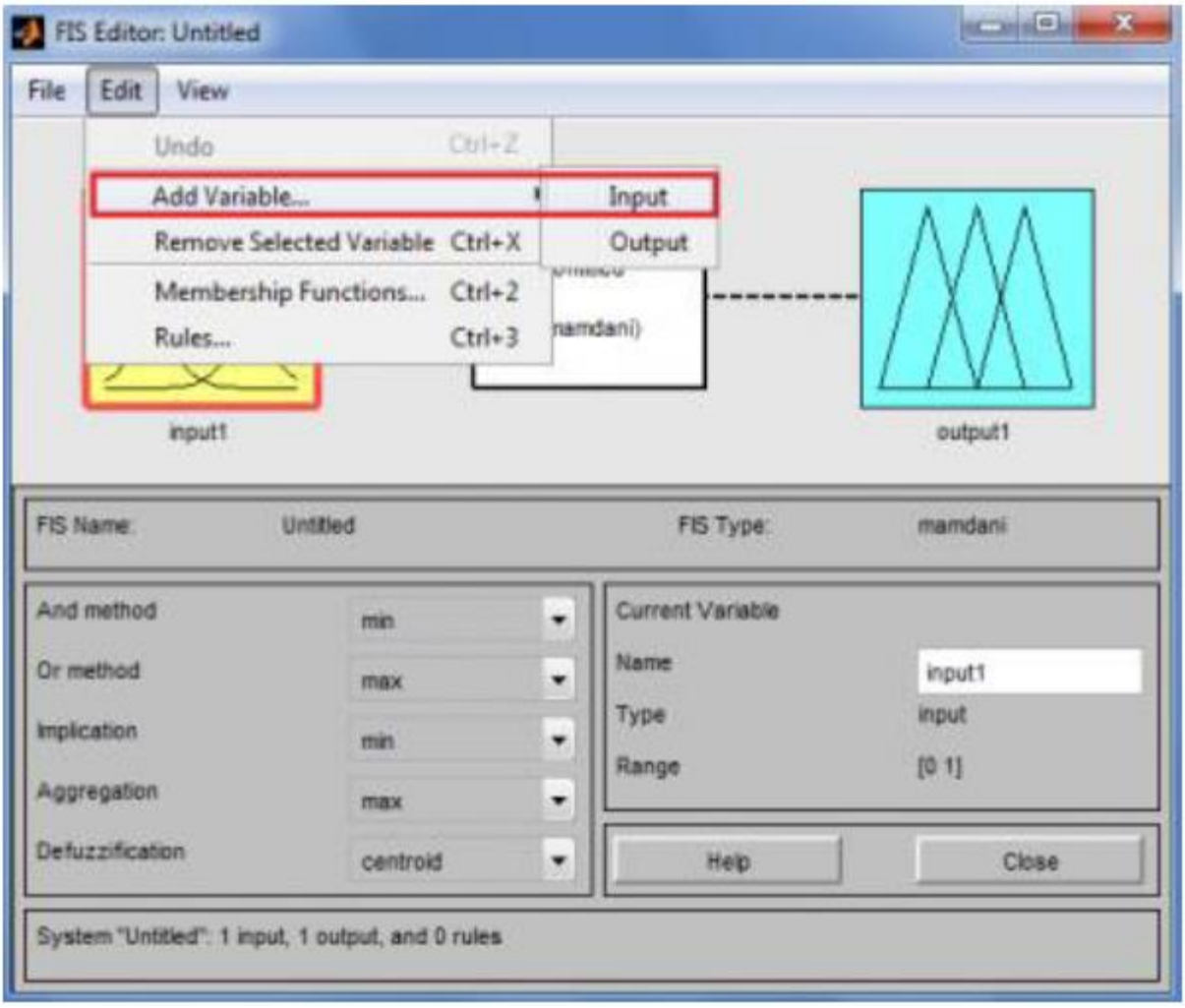
1. **Jika PELAYANAN Jelek ATAU MAKANAN Tengik, maka TIP Murah**
2. **Jika PELAYANAN Sedang, maka TIP Standar**
3. **Jika PELAYANAN Bagus ATAU MAKANAN Lezat, maka TIP Mahal**

Dapat kita simpulkan bahwa pasangan input-output adalah sebagai berikut:

INPUT: **PELAYANAN** (Jelek, Sedang, Bagus) dan **MAKANAN** (Tengik, Lezat)

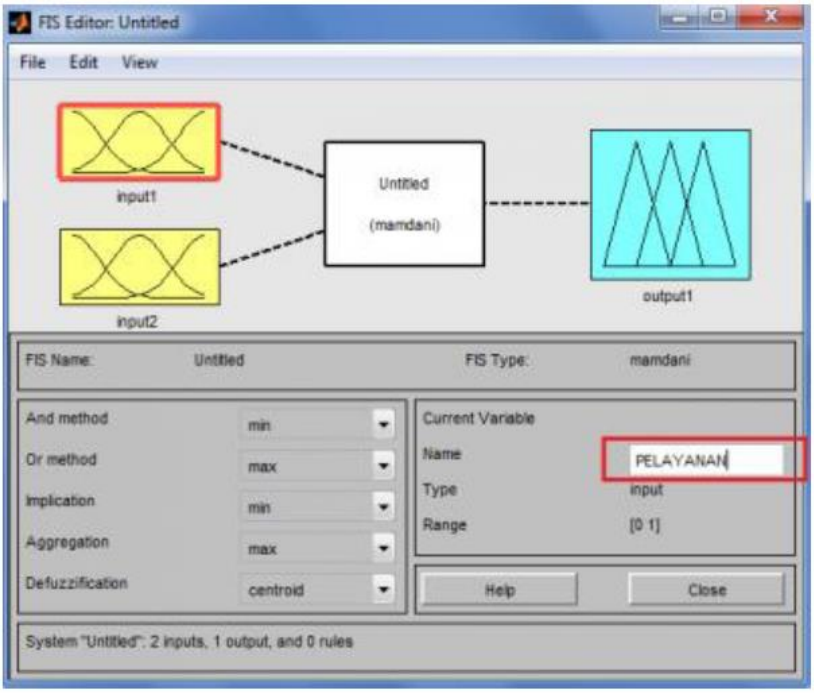
OUTPUT: **TIP** (Murah, Standar, Mahal)

Tampilan awal, FIS Editor secara default hanya menampilkan satu input dan satu output. Oleh karena itu, mari kita tambahkan satu buah input lagi. Pada **FIS Editor**, carilah **Edit**, kemudian **Add Variable...** dan klik **Input.**



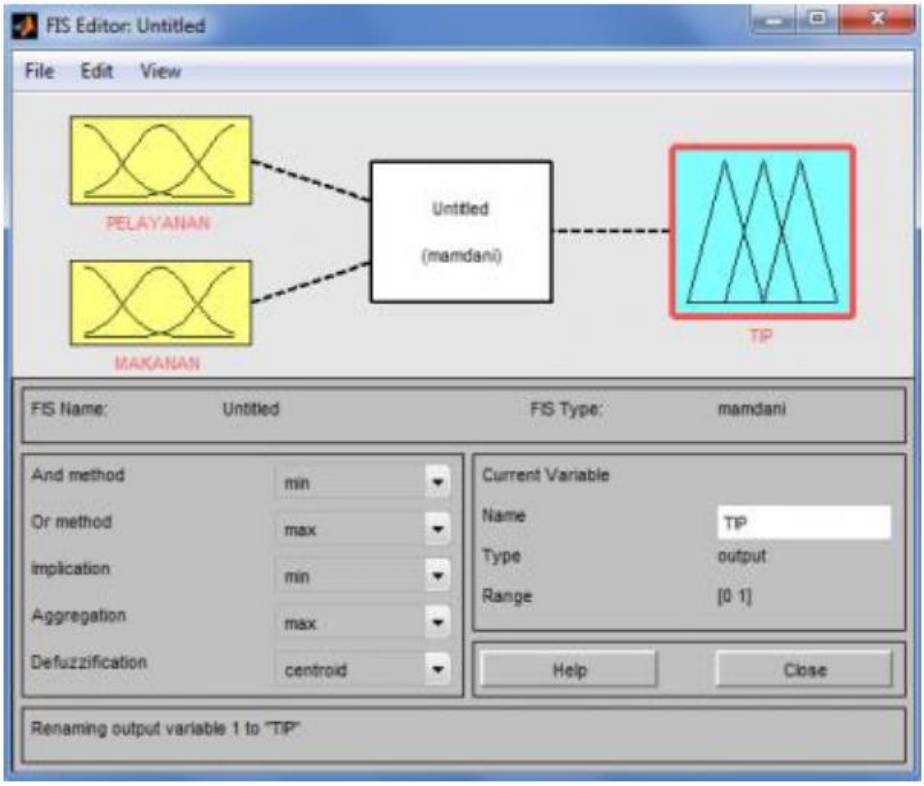
Gambar 7.19 Tampilan awal, FIS Editor secara default hanya menampilkan satu input dan satu output

Klik-lah input 1, kemudian ubahlah kolom **Name** pada **Current Variabel**. Gantilah namanya menjadi PELAYANAN, kemudian tekan Enter. Untuk parameter yang lain (And method, Or method, Implication, dan sebagainya) biarkanlah pada nilai defaulft-nya.



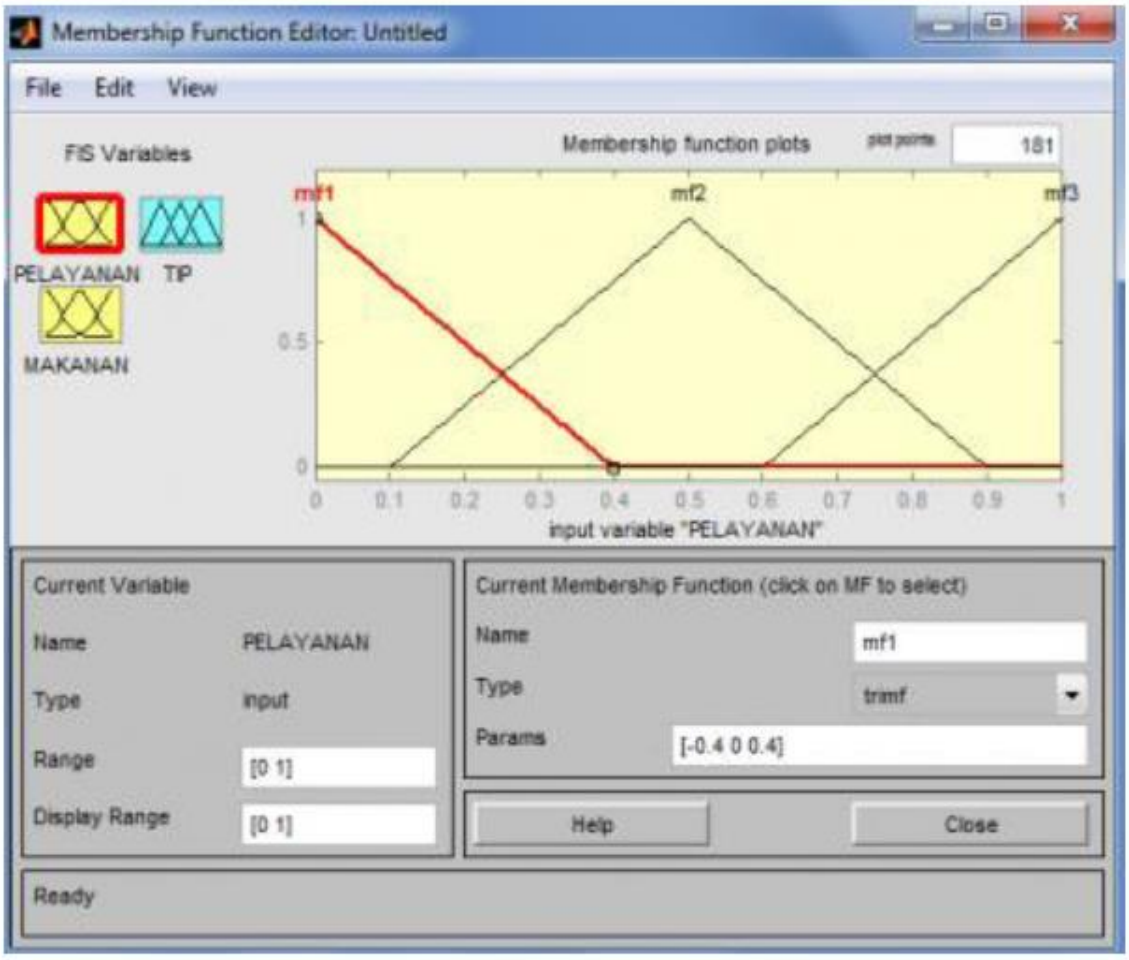
Gambar 7.20 Tampilan setelah, FIS Editor diedit

Lakukan hal yang sama untuk Input 2 dan Output, masing-masing dengan nama MAKANAN dan TIP. Hasilnya :



Gambar 7.21 Tampilan setelah, FIS Editor diedit variabel input dan output

Langkah selanjutnya adalah menentukan fungsi keanggotaan dari input dan output fuzzy. Mari kita lakukan perubahan pada input PELAYANAN terlebih dahulu. Klik-lah dua kali pada **kotak kuning** PELAYANAN. Selanjutnya akan muncul jendela **Membership Function Editor.**



Gambar 7.22 Tampilan variabel input

Pada Jendela ini, pastikan saat ini kotak kuning PELAYANAN masih aktif (ditandai dengan garis merah tebal). Klik-lah garis **mf1**, dan ubahlah parameter dibawahnya (ditunjukkan dengan kolom yang berwarna putih) sesuai dengan tabel berikut :

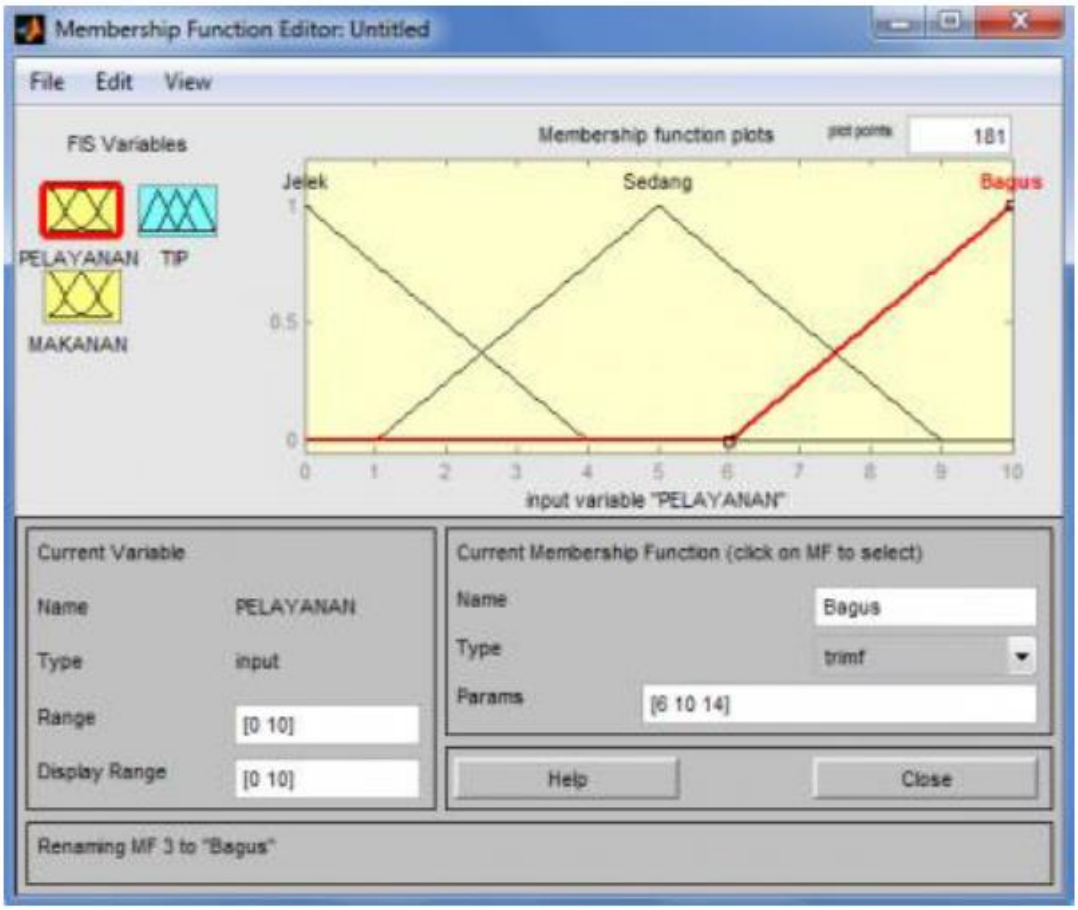
Parameter	**Nilai**
Range	[0 10]
Name	Jelek
Type	Trimf
Params	[-4 0 4]

Jika mf1 sudah diubah, lakukan langkah yang sama untuk mf2 dan mf3, masing masing seperti pada Tabel 2 dan 3.

Parameter	**Nilai**
Range	[0 10]
Name	Sedang
Type	Trimf
Params	[1 5 9]

Parameter	**Nilai**
Range	[0 10]
Name	Bagus
Type	Trimf
Params	[6 10 14]

Pada input PELAYANAN, semua fungsi keanggotaan memiliki tipe trimf (Triangular Membership Function) Yaitu fungsi keanggotaan berbentuk segitiga.



Gambar 7.23 Tampilan variabel input Pelayanan yang sudah dikasih nilai

Selanjutnya dilakukan penyesuaian untuk variabel MAKANAN dan TIP sama seperti mengubah variabel PELAYANAN diatas.

Fungsi keanggotaan MAKANAN: TENGIK

Parameter	**Nilai**
Range	[0 10]
Name	Tengik
Type	Trapmf
Params	[0 0 1 3]

Fungsi keanggotaan MAKANAN: LEZAT

Parameter	**Nilai**
Range	[0 10]
Name	Lezat

Type	Trapmf
Params	[7 9 10 10]

Fungsi keanggotaan TIP: MURAH

Parameter	**Nilai**
Range	[0 30]
Name	Murah
Type	Trimf
Params	[0 5 10]

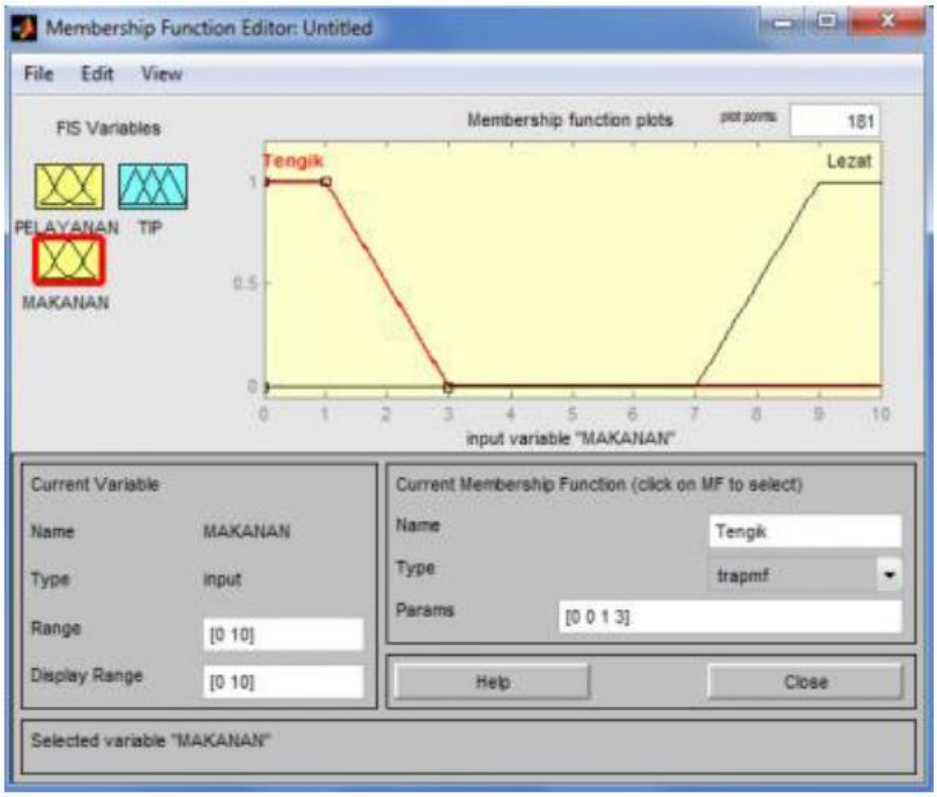
Fungsi keanggotaan TIP: STANDAR

Parameter	**Nilai**
Range	[0 30]
Name	Standar
Type	Trimf
Params	[10 15 20]

Fungsi keanggotaan TIP: MAHAL

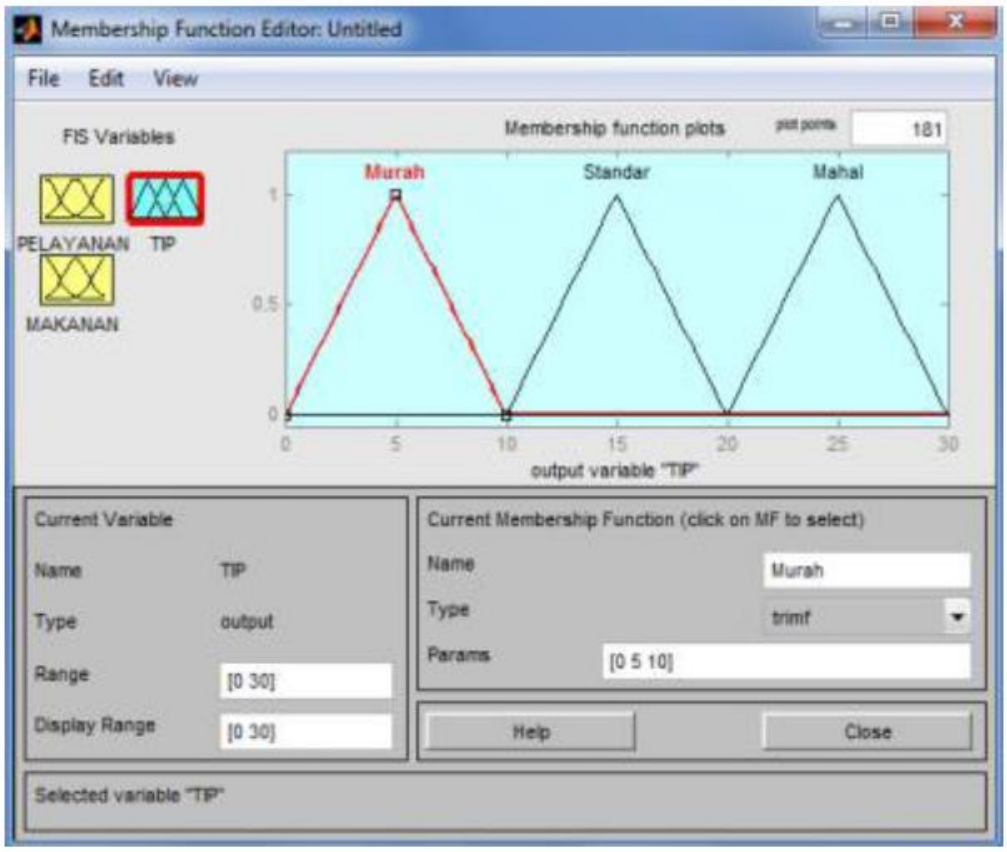
Parameter	**Nilai**
Range	[0 30]
Name	Mahal
Type	Trimf
Params	[20 25 30]

Hasil akhir dari fungsi keanggotaan MAKANAN adalah :



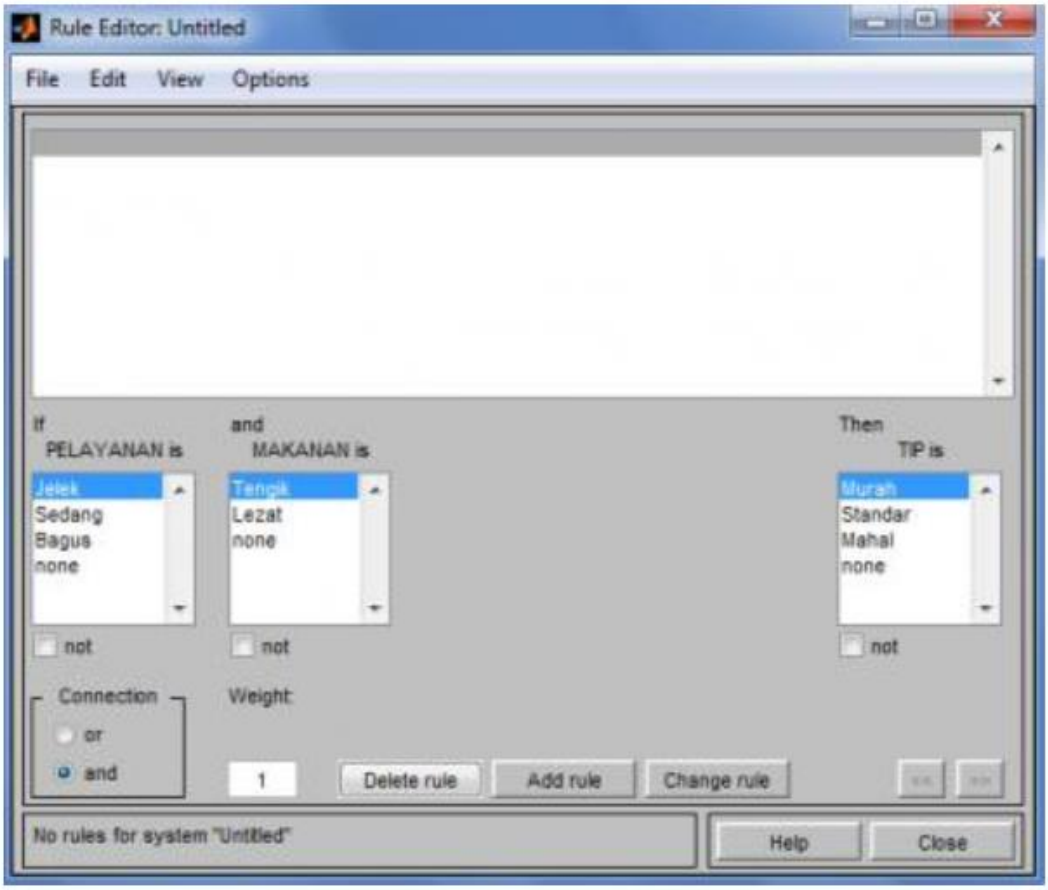
Gambar 7.24 Tampilan variabel input Makanan

Hasil akhir dari fungsi keanggotaan TIP adalah :



Gambar 7.25 Tampilan variabel Output TIP

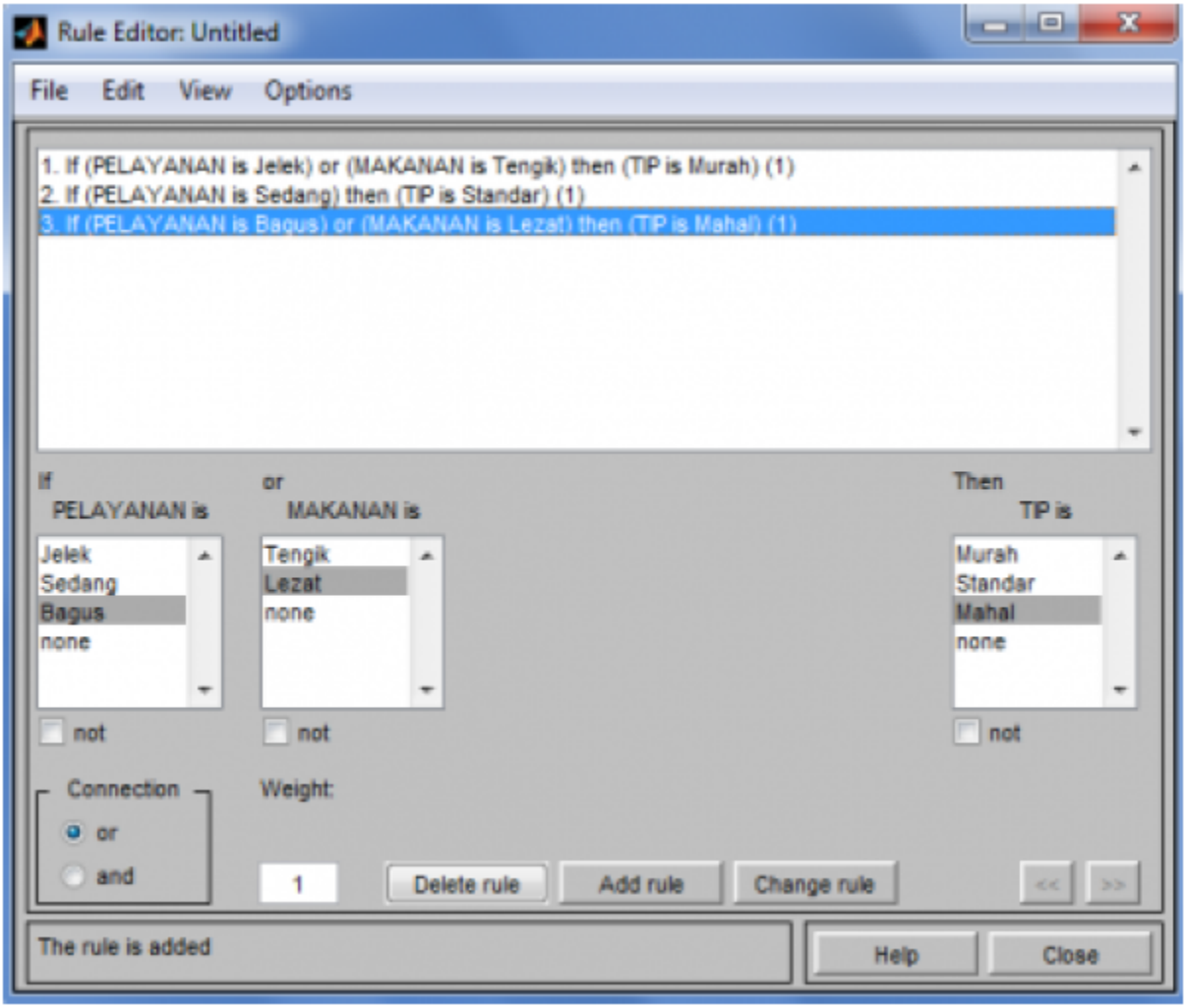
Langkah selanjutnya adalah menetapkan aturan-aturan menggunakan mekanisme **IF-THEN.** Masih pada jedela yang sama (Membership Function Editor), bukalah **Edit** kemudian klik **Rules...** . selanjutnya akan tampil jendela **Rule Editor.**



Gambar 7.26 Tampilan Rule editor

aturan-aturan yang berlaku pada contoh studi kasus ini, adalah :
1. Jika PELAYANAN Jelek **ATAU** MAKANAN Tengik, maka TIP Murah
2. Jika PELAYANAN Sedang, maka TIP Standar
3. Jika PELAYANAN Bagus **ATAU** MAKANAN Lezat, maka TIP Mahal

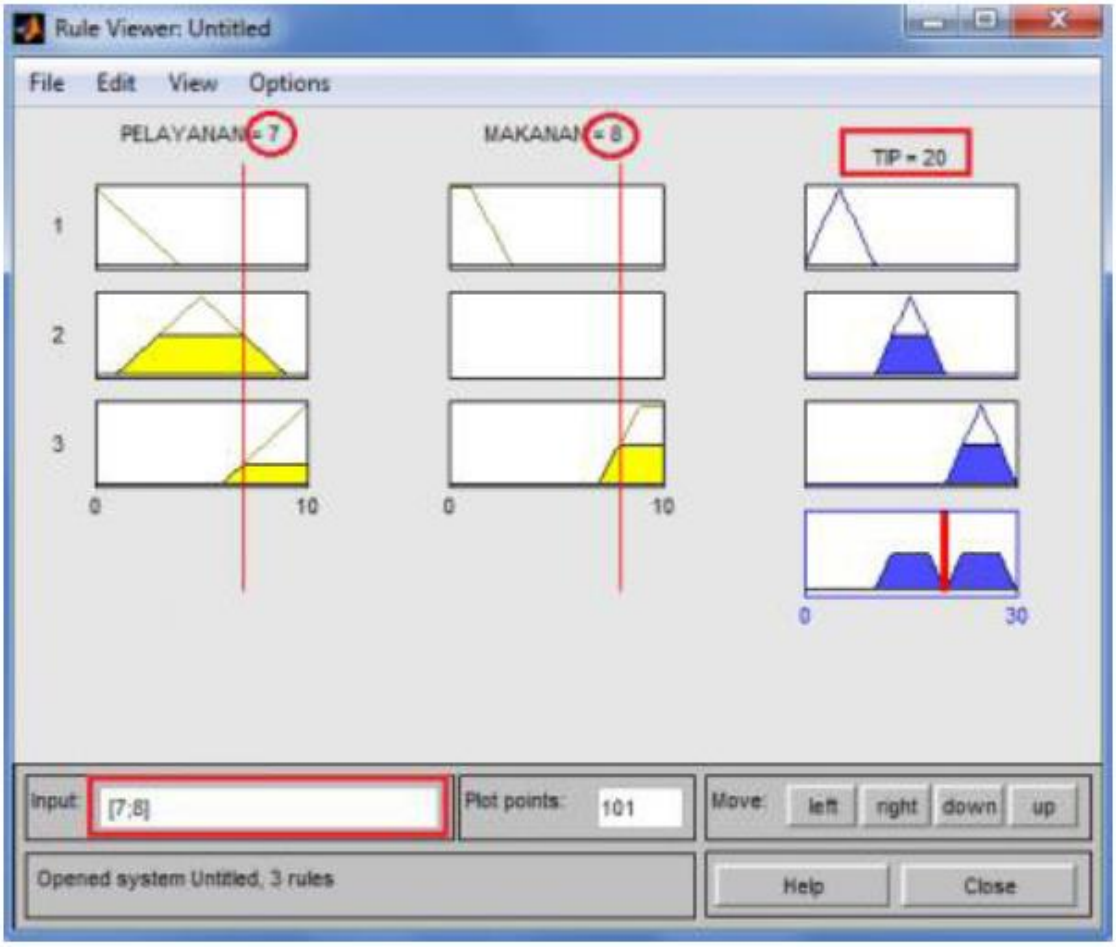
Selanjutnya pengaturan jendela Rule Editor disesuaikan dengan aturan diatas. Ubah-ubahlah kolom PELAYANAN, MAKANAN dan TIP, serta sesuaikan dengan pilihan Connection-nya (dalam hal ini OR atau AND). Dalam studi kasus ini koneksi menggunakan kata 'ATAU' sehingga menggunakan koneksi OR. Pada aturan kedua, terlihat hanya input PELAYANAN saja yang masuk dalam aturan, sedangkan MAKANAN tidak. Oleh karena itu, kita bisa pilih kolom MAKANAN dengan none. Jika telah selesai, klik-lah tombol Add Rule.
Kita juga bisa mengedit atau menghapus aturan yang kita buat dengan tombol yang telah disediakan.



Gambar 7.27 Tampilan Rule editor yang telah diedit

Apabila sudah, jendela Rule Editor dan Membership Function bisa ditutup, dan bisa kembali ke jendela awal. Di jendela utama, bisa dilihat hasil dari sistem fuzzy yang telah dibuat. Carilah **View**, kemudian klik **Rules**.

Seperti yang telah dibahas sebelumnya bahwa input diberi nilai tertentu, yaitu PELAYANAN = 7, dan MAKANAN = 8. Ubahlah kolom Input, dan lihat hasilnya. Terlihat bahwa nilai TIP = 20. Dengan demikian, dapat disimpulkan bahwa TIP yang harus dibayarkan oleh Romi adalah sebesar 20%. Nilai input ini dapat diubah-ubah dengan cara mengetikkan, atau menggeser-geser garis vertikal yang melintasi input.



Gambar 7.28 Tampilan Rule editor yang bisa diedit

Bentuk dari berbagai peluang nilai input dapat dilihat hasilnya dengan cara memilih **View - Surface**. Simpanlah projek ini dengan cara memilih **File - Export - To File...** Simpanlah sesuai dengan nama yang diinginkan. Hindari menggunakan spasi dalam penamaan.

7.4 Penelitian yang berhubungan dengan logika fuzzy

Penelitian ini menggunakan metode logika *fuzzy tsukamoto* untuk mengukur seberapa kecanduan seorang bermain game ini. Data yang diambil dari beberapa pengguna game *PlayerUnknown's Battlegrounds Mobile* (*PUBG Mobile*) kemudian dianalisis dengan menggunakan metode logika *fuzzy tsukamoto* dan dilakukan perhitungan guna mendapatkan kesimpulan yang tepat dan sesuai dengan yang diharapkan. Dari 100 data yang dilakukan mempunyai *output* denagn kecanduan rendah sebesar 50%, kecanduan sedang sebesar 20% dan kecanduan tinggi sebesar 30%.

Analisis Kecanduan *Game Player Unknown's Battlegrounds* (PUBG) *Mobile* dengan Menggunakan Logika *Fuzzy*

Mochammad Ilham Rofiqi[1], Hindarto Hindarto[2]

[1,2] Jurusan Informartika, Fakultas Sains dan Teknologi, Universitas Muhammadiyah Sidoarjo
hindarto@umsida.ac.id

Abstrak

Perkembangan game saat ini sangatlah pesat khususnya *game mobile*. Game dengan *game play* sederhana, singkat, dan mudah diakses lebih sering dimainkan dibanding *game* yang menawarkan permainan kompleks. Salah satunya ialah game *PlayerUnknown's Battlegrounds Mobile* (PUBG *Mobile*) game yang sudah di unduh mencapai 400 juta pada tahun 2019 dan pengguna harian game ini mencapai 50 juta pengguna. Dengan menggunakan metode logika *fuzzy tsukamoto* untuk mengukur seberapa kecanduan seorang bermain game ini. Data yang diambil dari beberapa pengguna game *PlayerUnknown's Battlegrounds Mobile* (*PUBG Mobile*) kemudian dianalisis dengan menggunakan metode logika *fuzzy tsukamoto* dan dilakukan perhitungan guna mendapatkan kesimpulan yang tepat dan sesuai dengan yang diharapkan. Dari 100 data yang dilakukan mempunyai *output* denagn kecanduan rendah sebesar 50%, kecanduan sedang sebesar 20% dan kecanduan tinggi sebesar 30%.

Kata Kunci – kecanduan game, logika fuzzy, tsukamoto, *player unknown's battlegrounds mobile*

1. Pendahuluan

Saat ini perkembangan kemajuan teknologi informasi semakin cepat dan pesat salah satunya ialah teknologi internet dan komputer (Ratnasari, 2004). Hampir setiap kalangan menggunakan kemajuan teknologi ini mulai dari yang tua sampai muda semua menggunakan jaringan internet. Dan internet juga mempunyai dampak negatif dan positif tergantung penggunannya. Bermacam-macam informasi mudah diperoleh secara langsung dari internet secara cepat. Bukan hanya itu saja, media hiburan mulai dari vidio ataupun foto juga mudah diakses oleh internet (Ngafifi, 2014).

Dari berbagai macam media hiburan yang mudah diakses *game online* juga sekarang mudah diakses secara bebas (Ismi & Akmal, 2020)(Putra et al., 2019). Pada saat ini permainan yang diakses dengan internet sedang marak maraknya dikalangan masyarakat terutama bagi anak-anak sampai remaja. Permainan yang diakses secara online secara umum diperuntukkan untuk mengusir kebosanan dan hanya sekedar untuk menyegarkan pikiran setelah melakukan aktifitas. Walaupun kenyataanya permainan online malah membuat orang-orang kecanduan memainkan game ini. seringnya bermain secara terus-terusan membuat seorang terkena kecanduan bermain game yang berdampak pada tingkah laku (Masya & Candra, 2016)(Ayu & Saragih, 2016).

Kecanduan game online membuat para pemain lebih banyak menghabiskan waktu didepan komputer ataupun handphone sehingga menghambat interaksi dengan sekitar. Dan juga game online mempunyai banyak dampak mulai dari kurangnya berdapatasi sosial yang mengakibatkan fungsi psikologi seorang dan masalah sosial antar lingkungan sekitar (Lutfiwati, 2018)(Putra et al., 2019).

Dari sekian banyak game online di internet salah satunya paling banyak dimainkan ialah game *PlayerUnknown's Battlegrounds Mobile* (PUBG Mobile) yang sudah diunduh sekitar 400 juta pada tahun 2019 dan pengguna harian sampai 50 juta merupakan game *battle royal* yang dimainkan secara online dan secara bersama-sama sehingga membuat pemain kecanduan game ini (Fauzi, 2019)(Rosalino Triyantama & Santoso, 2019). Dengan mengguunakan metode logika fuzzy tsukamoto untuk mengukur tingkat kecanduan seorang bermain game *PlayerUnknown's Battlegrounds Mobile* (PUBG Mobile). Dengan memanfaatkan data para pemain game ini untuk mengukur tingkat kecanduan.

Pada penelitian ini, metode yang digunakn menggunkan metode logika *Fuzzy*. Logika *fuzzy* merupakan metode atau cara yang bisa digunakan untuk mendeteksi bebrbagai macam kegiatan, contohnya mendeteksi kapan terjadinya banjir (Arifin et al., 2016). Logika *fuzzy* juga bisa digunkan untuk berbagai hal yang lain (Sugianti et al., 2020)(Maulana et al., 2018)(Fiano & Purnomo, 2017).

Tujuan penelitian ini yaitu untuk membangun sistem untuk mendeteksi kecanduan bermain game *PlayerUnknown's Battlegrounds Mobile*. Beberapa pembahasan yang akan disampaikan, diataranya pendahuluan, metode penelitian, hasil dan pembahasan.

Soal :

1. Buatlah Fungsi/derajat keanggotaan seseorang yang mempunyai umur 50 tahun. Dengan kurva – S
2. Buatlah Fungsi/derajat keanggotaan suhu yang mempunyai temperatur 50 derajat C. Dengan
    a. **Representasi Linear Naik**
    b. Representasi Kurva Segitiga
    c. Representasi Kurva Trapesium

3. Studi Kasus

Doni ingin mengajak Juli dinner di suatu restoran. Romi ingin membagi kebahagiaan pada malam ini kepada pelayan restoran dengan memberikan uang tip. Dia akan memberikan uang tip sebesar 1**5-30%** dari total belanjanya. Besarnya uang tip diberikan berdasarkan tingkat **PELAYANAN** dan kualitas **MAKANAN** yang dihidangkan.

Bantulah Doni untuk memutuskan besarnya uang tip, jika setelah menikmati hidangan dan fasilitas PELAYANAN Doni memberi nilai:

PELAYANAN = 8

MAKANAN = 7

Penilaian PELAYANAN dan MAKANAN berada pada rentang nilai 0-10 dan kisaran uang tip adalah 15 - 30%.

Adapun aturan pemberian tip yg ditetapkan oleh Romi adalah sebagai berikut:

1. **Jika PELAYANAN Jelek ATAU MAKANAN Tengik, maka TIP Kecil**
2. **Jika PELAYANAN Sedang, maka TIP Standar**
3. **Jika PELAYANAN Bagus ATAU MAKANAN Lezat, maka TIP Besar**

BAB 8
Jaringan Syaraf Tiruan

Jaringan Syaraf Tiruan yang disingkat dengan nama JST merupakan suatu sistem informasi yang diapdosi oleh sistem syaraf biologi, seperti sistem syaraf yang berda di otak manusia. Sistem ini merupakan sistem baru yang digunakan untuk pemrosesan dari sebuah sistem informasi. Sistem ini terdiri dari beberapa neuran yang saling berhubungan satu dengan yang lainnya, sehingga menghasilkan sebuah sistem informasi. Jaringan syaraf ini digunakan untuk sistem yang bisa diapdosi dari seorang manusia, sehingga ada proses pembelajan didalamnya. JST dapat digunakan dalam berbagai aplikasi tertentu seperti halnya pengenakan sidik jari, pengenalan wajah, pengenalan pola dan semua sistem yang bekerja berdasarkan proses pembelajaran. Pemebelajaran di JST ini, melelui proses yang ada di tiap-tiap neuran yang berhubungan satiu dengan neuron yang lainnya.

Simulasi jaringan saraf tampaknya merupakan perkembangan baru-baru ini. Tetapi, simulasi jaringan syaraf berdiri sebelum perangkat komputer ada. Banyak minat dan kemajuan telah didorong oleh penggunaan emulasi komputer yang murah. Setelah periode awal antusiasme, bidang ini selamat dari periode frustrasi dan reputasi buruk. Selama periode ini ketika pendanaan dan dukungan profesional sangat minim, kemajuan yang dicapai oleh para peneliti relatif sedikit. Para pionir ini mampu mengembangkan teknologi menarik yang melampaui batasan yang diidentifikasi oleh Minsky dan Papert. Minsky dan Papert, menerbitkan sebuah buku (1969) di mana mereka merangkum perasaan umum frustrasi (terhadap jaringan saraf) di antara para peneliti, dan dengan demikian diterima oleh sebagian besar tanpa analisis lebih lanjut. Saat ini, bidang jaringan saraf menikmati kebangkitan minat dan peningkatan pendanaan yang sesuai. Ahli neurofisiologi Warren McCulloch dan ahli logika Walter Pits tahun 1943 telah memproduksi neuron buatan pertama. Namun pengetahuan dan teknologi pada saat itu tidak memungkinkan mereka dapat melakukan sesuatu yang banyak. Jaringan saraf, dapat digunakan untuk mengekstrak pola dan mendeteksi sesuatu yang kompleks, sehingga dapat digunakan untuk mengetahui data yang sangat rumit. Sistem JST yang sudah terlatih dapat digunakan sebagai pakar dalam hal sebagai informasi untuk memperoleh data yang sesuai dan dapat dianalisa. Sistem ini dapat digunakan untuk keperluan memperediksi atau mengolah data yang kita inginkan untuk memperoleh informasi baru. Keuntungan lainnya termasuk:

1. Pembelajaran adaptif: Kemampuan dari JST untuk melaksanakan tanggung jawabnya terhadap data yang diberikan, sehingga dapat dilakukan untuk proses pengenalan dan pelatihan.
2. Self-Organization: JST selama proses pembelajaran dapat melakukan representasi atau membuat organisasinya sendiri dari informasi yang didapatkan.
3. Operasi Waktu Nyata: kemampuan yang diproduksi dan dimanfaatkan JST dalam hal menghitung secara pararel dengan perangkat keras khusus.
4. Toleransi Kesalahan melalui Pengodean Informasi yang Berlebihan: Penghancuran sebagian jaringan menyebabkan penurunan kinerja yang sesuai. Tetapi kemampuan untuk mempertahankan dari jaringan yang mengalami kerusakan sangat besar.

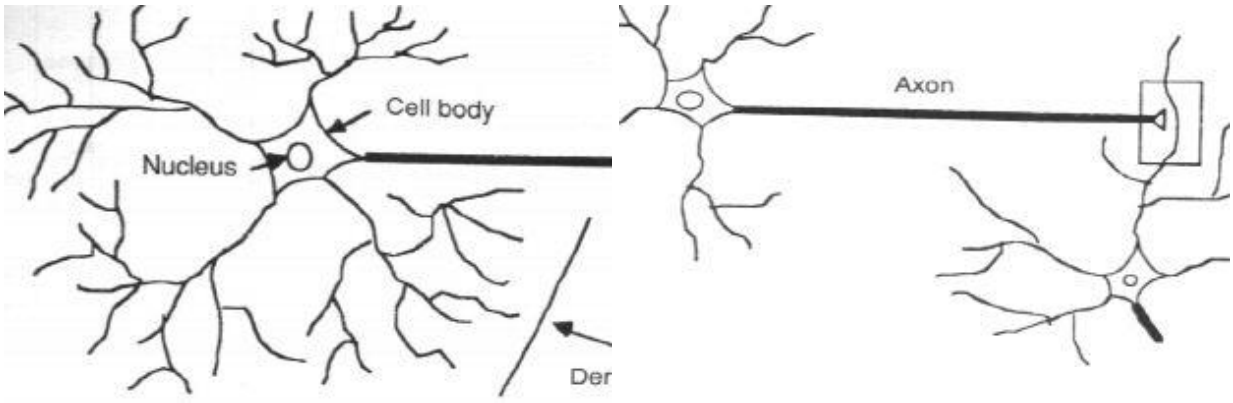
8.1 Jaringan saraf versus komputer konvensional

Untuk memecahkan masalah Jaringan saraf mengambil pendekatan yang berbeda dari pada Komputer konvensional. Komputer konvensional mengikuti serangkaian instruksi dalam menyelesaikan permasalahan yang dinamakan pendekatan secara algoritmik. Kecuali jika langkah-langkah spesifik yang perlu diikuti komputer diketahui, komputer tidak dapat menyelesaikan masalah. Itu membatasi kemampuan pemecahan masalah komputer konvensional pada masalah yang sudah kita pahami dan ketahui bagaimana menyelesaikannya. Tetapi komputer akan jauh lebih berguna jika mereka dapat mengerti apa yang mereka lakukan dalam hal yang kita tidak tahu persis bagaimana untuk melakukannya.

Jaringan saraf mempunyai cara yang sama seperti yang dilakukan otak manusia dalam melakukan proses informasi. Jaringan ini mempunyai neuron-neuron yang besar dan saling berhubungan setra bekerja secara pararel dalam menangani suatu permasalahan. Jaringan saraf belajar dengan memberi contoh. Mereka tidak dapat diprogram untuk melakukan tugas tertentu. Contoh harus dipilih dengan hati-hati jika tidak, waktu yang berguna terbuang sia-sia atau bahkan lebih buruk lagi, jaringan mungkin tidak berfungsi dengan benar. Kerugiannya untuk memecahkan masalah dengan sendirinyajaringan mempunyai cara sendiri, sehingga dalam hal pengoperasian mereka tidak dapat diprediksi. Dilain pihak, untuk memecahkan masalah komputer konvensional mempunyai pendekatan secara kognitif. Yaitu cara pemecahan masalah harus diketahui dan dinyatakan dalam instruksi kecil yang tidak ambigu. Instruksi ini kemudian diubah menjadi program bahasa tingkat tinggi dan kemudian menjadi kode mesin yang dapat dimengerti oleh komputer. Kode mesin ini benar-benar dapat diprediksi, jika ada yang salah karena adanya kesalahan perangkat lunak atau perangkat keras.
Jaringan saraf dan komputer algoritmik konvensional mempunyai tindakan saling melengkapi dan tidak terjadi persaingan. Ada tugas yang lebih cocok untuk pendekatan algoritmik seperti operasi aritmatika dan tugas yang lebih cocok untuk jaringan saraf. Terlebih lagi, terdapat tugas besar yaitu sistem yang melakukan kombinasi dari dua pendekatan sehingga dihasilkan pekerjaan yang lebih efisien dan dapat memaksimalkan proses.

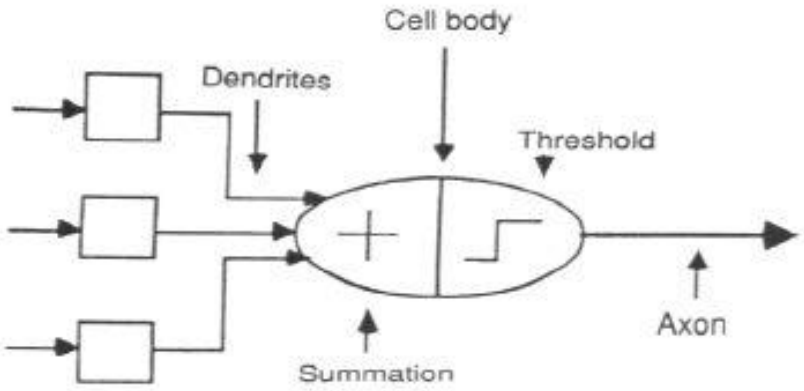
8.2 Neuron Manusia dan Buatan - menyelidiki kesamaan

Masih banyak yang belum diketahui tentang bagaimana otak melatih dirinya untuk memproses informasi, sehingga banyak teori. Di otak manusia, neuron tipikal mengumpulkan sinyal dari orang lain melalui sejumlah struktur halus yang disebut dendrit. Neuron mengirimkan lonjakan aktivitas listrik melalui stand panjang dan tipis yang dikenal sebagai akson, yang terbagi menjadi ribuan cabang. Pada akhir setiap cabang, struktur yang disebut sinapsis mengubah aktivitas dari akson menjadi efek listrik yang menghambat atau merangsang aktivitas dari akson menjadi efek listrik yang menghambat atau menggairahkan aktivitas di neuron yang terhubung. Ketika neuron menerima masukan rangsang yang cukup besar dibandingkan dengan masukan penghambatannya, ia mengirimkan lonjakan aktivitas listrik ke aksonnya. Pembelajaran terjadi dengan mengubah keefektifan sinapsis sehingga pengaruh satu neuron terhadap neuron lainnya berubah.



Gambar 8.1 Komponen neuron dan sinapsis

Dari Neuron Manusia ke Neuron Buatan
Peneliti melakukan jaringan saraf ini dengan terlebih dahulu mencoba menyimpulkan fitur penting dari neuron dan interkoneksinya. Peneliti kemudian biasanya memprogram komputer untuk mensimulasikan

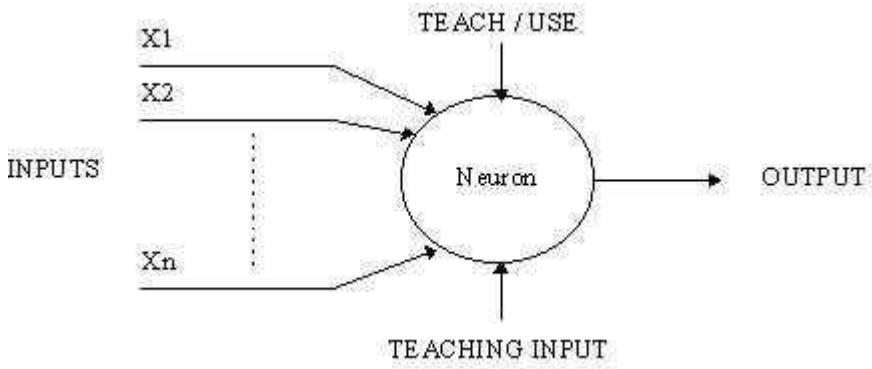


Gambar 8.2 Model neuron

fitur-fitur ini. Namun karena pengetahuan kita tentang neuron tidak lengkap dan daya komputasi kita terbatas, model kita tentu merupakan idealisasi kasar dari jaringan neuron yang sebenarnya.

8.3 . Pendekatan rekayasa

Neuron buatan adalah perangkat dengan banyak input dan satu output. Neuron memiliki dua mode operasi; mode pelatihan dan mode penggunaan. Dalam mode pelatihan, neuron dapat dilatih untuk menembak (atau tidak), untuk pola input tertentu. Untuk mode penggunaan, bial pola dari sebuah inputan yang diajarkan terdeteksi pada input, maka output terkaitnya menjadi sebuah output saat ini. Bila pola input tidak termasuk dalam daftar pola input yang telah dilatih, maka aturan pengaktifan dapat digunakan untuk menentukan apakah akan aktif atau tidak.



Gambar 8.3 Neuron sederhana

8.4 Firing rules

Aturan penembakan (Firing rules) adalah konsep penting dalam jaringan saraf dan memperhitungkan fleksibilitasnya yang tinggi. Aturan pengaktifan menentukan bagaimana seseorang menghitung apakah neuron harus diaktifkan untuk pola input apa pun. Ini terkait dengan semua pola input, tidak hanya pola yang digunakan node tersebut untuk dilatih. Aturan menembak sederhana dapat diterapkan dengan menggunakan teknik jarak Hamming. Aturannya sebagai berikut: Ambil kumpulan pola pelatihan untuk sebuah node, beberapa di antaranya menyebabkannya menyala (set pola yang diajarkan 1) dan lainnya yang mencegahnya melakukannya (set yang diajarkan 0). Apabila sebuah pola yang belum terdapat pola yang diinputkan, maka menyebabkan node akan menyala. Untuk perbandingan, pola tersebut memiliki

lebih banyak elemen input yang sama dengan pola 'terdekat' maka akan di set 1 dengan pola 'terdekat' di set 0-train. Jika ada, maka polanya tetap dalam keadaan tidak terdefinisi.
Misalnya, 3 input neuron akan diajarkan dalam menghasilkan 1 jika inputan (A1,A2 dan A3) mempunyai output 111 atau 101 dan mempunyai output 0 jika inputnya 000 atau 001. Berikut ini adalah tabel kebenaran untuk contoh.

Tabel 8.1 Tabel kebenaran tiga input neuron

A1:		0	0	0	0	1	1	1	1
A2:		0	0	1	1	0	0	1	1
A3:		0	1	0	1	0	1	0	1
Output		0	0	0/1	0/1	0/1	1	0/1	1

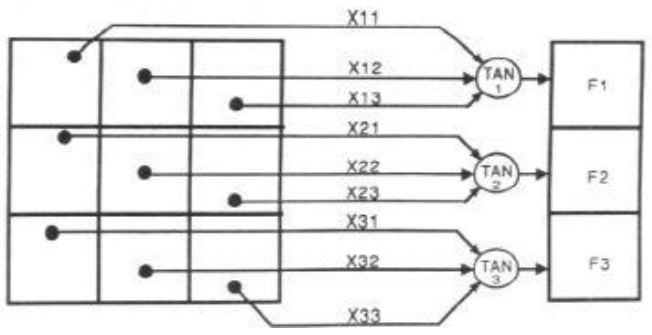
Sebagai contoh penerapan aturan pengaktifan, ambil pola 010. Ini berbeda dari 000 dalam 1 elemen, dari 001 dalam 2 elemen, dari 101 dalam 3 elemen, dan dari 111 dalam 2 elemen. sehingga, pola 'terdekat' yaitu 000 termasuk dalam himpunan 0 yang diajarkan. Jadi aturan pengaktifan mensyaratkan bahwa neuron tidak boleh menyala ketika inputnya 001. Untuk kondisi lain, 011 mempunyai pola yang jauh dari dua pola yang telah diajarkan dan memiliki output berbeda untuk itu output tetap tidak terdefinisi (0/1).
Dengan menerapkan penembakan di setiap kolom diperoleh tabel kebenaran sebagai berikut;
Tabel 8.2 Tabel kebenaran hasil firing rules

A1:		0	0	0	0	1	1	1	1
A2:		0	0	1	1	0	0	1	1
A3:		0	1	0	1	0	1	0	1
Output		0	0	0	0/1	0/1	1	1	1

Selisih diantara kedua tabel kebenaran merupakan proses generalisasi neuron. Untuk itu aturan pengaktifan memberi neuron rasa kesamaan dan memungkinkannya untuk merespon 'secara masuk akal' terhadap pola yang tidak terlihat selama pelatihan.

8.5 Pengenalan Pola

Aplikasi penting dari jaringan saraf adalah pengenalan pola. Pengenalan pola dapat diimplementasikan dengan menggunakan jaringan saraf feed-forward (gambar 8.4) yang telah dilatih sesuai dengan itu. Selama pelatihan, jaringan dilatih untuk mengasosiasikan output dengan pola input. Ketika jaringan digunakan, ia mengidentifikasi pola input dan mencoba mengeluarkan pola output terkait. Kekuatan jaringan saraf menjadi hidup ketika sebuah pola yang tidak memiliki output yang terkait dengannya, diberikan sebagai input. Dalam hal ini, jaringan memberikan output yang sesuai dengan pola input yang diajarkan yang paling tidak berbeda dari pola yang diberikan.



Gambar 8.4 jaringan saraf feed-forward

Sebagai contoh:

Gambar 8.4 jaringan mengetahui untuk mengenali pola T dan H. Pola terkait semuanya hitam dan putih masing-masing seperti yang ditunjukkan di bawah ini.

Gambar 8.5. pola T dan H

Kotak hitam dipresentasikan dengan nilai 0 dan utnutk kotak putih dipresentasikan dengan nilai 1, sehingga tabel yang terbentuk adalah sebagai berikut :

Tabel 8.3 Tabel kebenaran untuk 3 neuron setelah generalisasi

A1:		0	0	0	0	1	1	1	1
A2:		0	0	1	1	0	0	1	1
A3:		0	1	0	1	0	1	0	1
Output		0	0	1	1	0	0	1	1

Top neuron

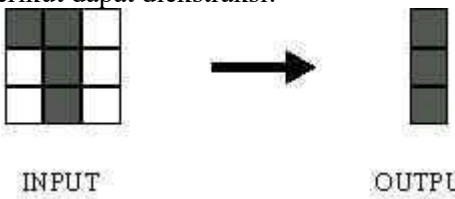
A1:		0	0	0	0	1	1	1	1
A2:		0	0	1	1	0	0	1	1
A3:		0	1	0	1	0	1	0	1
Output		1	0/1	1	0/1	0/1	0	0/1	0

Middle neuron

A1:		0	0	0	0	1	1	1	1
A2:		0	0	1	1	0	0	1	1
A3:		0	1	0	1	0	1	0	1
Output		1	0	1	1	0	0	1	0

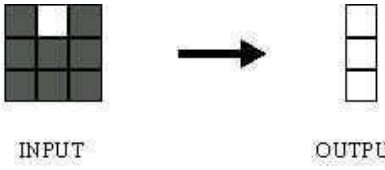
Bottom neuron

Dari tabel dapat dilihat asosiasi berikut dapat diekstraksi:



Gambar 8.6. pola input dan output dari tabel 8.3

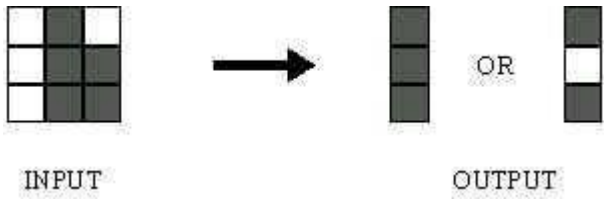
Untuk output harus yang berwarna hitam semua karena umtuk pola inputnya hampir sama dengan pola huruf 'T'.



Gambar 8.7. Pola inputan yang hampir sama dengan pola huruf 'T'

Untuk outputan harus putih semua karena untuk pola inputan hampir sama dengan pola huruf 'H'

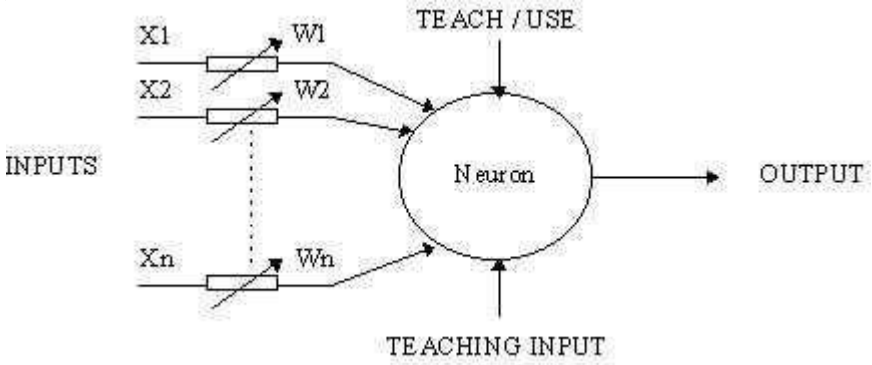
dalam hal ini, baris teratas berjarak 2 kesalahan dari T dan 3 dari H. Jadi output teratas berwarna hitam. Baris tengah berjarak 1 kesalahan dari T dan H sehingga outputnya acak. Baris bawah berjarak 1 kesalahan dari T dan 2 kesalahan dari H. Oleh karena itu, keluarannya berwarna hitam. Total output jaringan masih mendukung bentuk T



Gambar 8.8. Total output jaringan masih mendukung bentuk T

8.6 Neuron yang lebih rumit

Neuron sebelumnya tidak melakukan apa pun yang tidak dilakukan oleh komputer konvensional konvensional. Neuron yang lebih canggih (gambar 8.9) adalah model McCulloch dan Pitts (MCP). Perbedaan dari model sebelumnya adalah bahwa inputnya 'berbobot', efek yang dimiliki setiap input pada pengambilan keputusan tergantung pada bobot input tertentu. Bobot suatu masukan adalah suatu bilangan yang bila dikalikan dengan masukan memberikan masukan berbobot. Input berbobot ini kemudian ditambahkan bersama-sama dan jika melebihi nilai ambang batas yang telah ditentukan sebelumnya, neuron akan menyala. Dalam kasus lain, neuron tidak menyala.



Gambar 8.9. Sebuah neuron MCP

Dalam istilah matematika, neuron menyala jika dan hanya jika;

$$X1W1 + X2W2 + X3W3 + ... > T$$

Penambahan bobot input dan ambang batas membuat neuron ini menjadi sangat fleksibel dan kuat. Neuron MCP memiliki kemampuan untuk beradaptasi dengan situasi tertentu dengan mengubah bobot dan/atau ambangnya. Berbagai algoritma ada yang menyebabkan neuron 'beradaptasi'; yang paling sering digunakan adalah aturan Delta dan propagasi kesalahan kembali. Yang pertama digunakan dalam jaringan umpan-maju dan yang terakhir dalam jaringan umpan balik

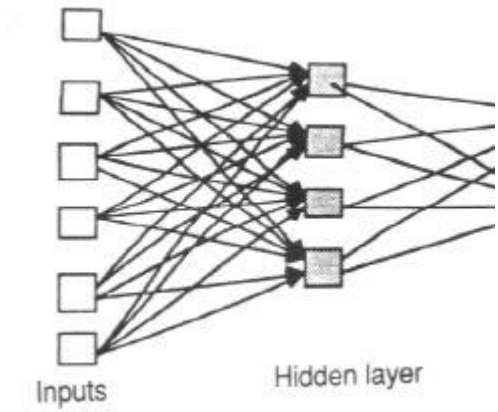
8.7 Arsitektur jaringan saraf

Jaringan umpan maju

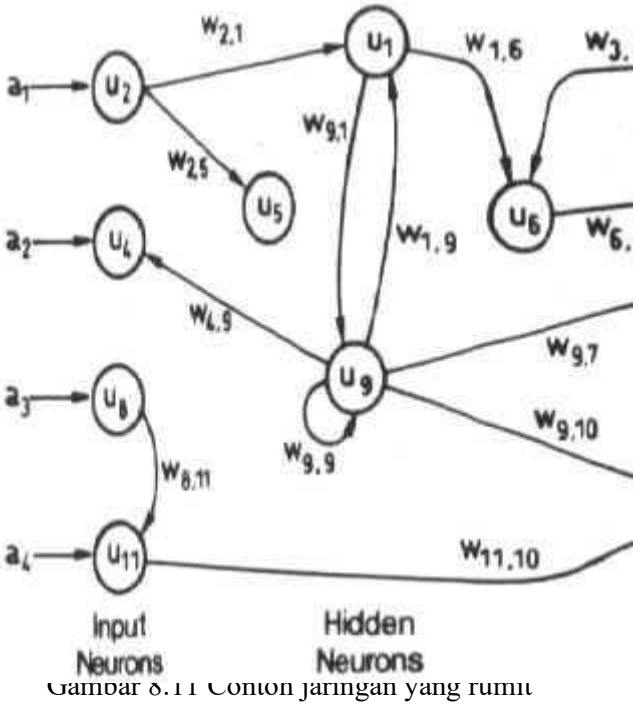
JST pada proses feed forward seperti pada gambar 1 dapat memberikan sinyal untuk berjalan satu arah pada masukan ke keluaran. Looping tidak terjadi yaitu pada setiap lapisan output tidak dapat mempengaruhi lapisan yang sama. JST feed forward merupakan proses dari jaringan lurus ke depan yang menghubungkan jaringan input ke jaringan output.

Jaringan umpan balik

Jaringan umpan balik seperti pada gambar 1 merupakan jaringan yang dapat memungkinkan proses jaringan jang dapat berjalan dari input ke output dan sebaliknya. Pada Jaringan umpan balik ini, prosesnya sangat rumit dan sangat besar. Pada jaringan umpan balik ini mempunyai karakter dinamis untuk menghasilkan output yang baik, sehingga prosesnya berjalan terus menerus sampai seimbang. Jaringan umpan balik ini mempunyai karakteristik tetap pada titik ekuilibrium, sehingga proses umpan balik ini mengijinkan input berubah terus dan keseimbangan baru perlu ditemukan. Arsitektur umpan balik juga disebut sebagai interaktif atau berulang, meskipun istilah yang terakhir sering digunakan untuk menunjukkan koneksi umpan balik dalam organisasi lapisan tunggal.



Gambar 8.10 Contoh jaringan feedforward sederhana



Gambar 8.11 Contoh jaringan yang rumit

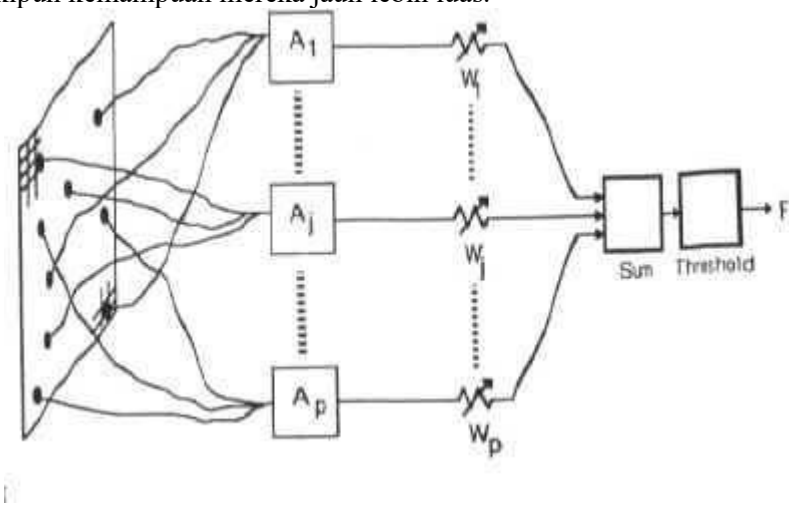
8.8 Lapisan jaringan

Jenis jaringan saraf tiruan yang paling umum terdiri dari tiga lapisan unit, pertama lapisan input, kedua ke lapisan tersembunyi, dan yang ketiga lapisan output seperti pada gambar 8.10. Aktivitas unit input mewakili informasi mentah yang dimasukkan ke dalam jaringan. Aktivitas setiap unit tersembunyi ditentukan oleh aktivitas unit input dan bobot pada hubungan antara input dan unit tersembunyi. Perilaku

unit keluaran tergantung pada aktivitas unit tersembunyi dan bobot antara unit tersembunyi dan unit keluaran.
Jenis jaringan sederhana ini menarik karena unit tersembunyi bebas untuk membangun representasi inputnya sendiri. Bobot antara input dan unit tersembunyi menentukan kapan setiap unit tersembunyi aktif, dan dengan memodifikasi bobot ini, unit tersembunyi dapat memilih apa yang diwakilinya.
Kami juga membedakan arsitektur single-layer dan multi-layer. Organisasi lapisan tunggal, di mana semua unit terhubung satu sama lain, merupakan kasus yang paling umum dan memiliki kekuatan komputasi yang lebih potensial daripada organisasi multi-lapisan yang terstruktur secara hierarkis. Dalam jaringan multi-lapisan, unit sering diberi nomor oleh lapisan, bukan mengikuti penomoran global.

8.9 Perceptron

Karya paling berpengaruh pada jaringan saraf di tahun 60-an berada di bawah judul 'perceptrons', sebuah istilah yang diciptakan oleh Frank Rosenblatt. Perceptron (gambar 8.12) ternyata menjadi model MCP (neuron dengan input berbobot) dengan beberapa tambahan, tetap, pra-pemrosesan. Unit berlabel A1, A2, Aj, Ap disebut unit asosiasi dan tugasnya adalah mengekstrak fitur spesifik yang dilokalkan dari gambar input. Perceptron meniru ide dasar di balik sistem visual mamalia. Mereka terutama digunakan dalam pengenalan pola meskipun kemampuan mereka jauh lebih luas.



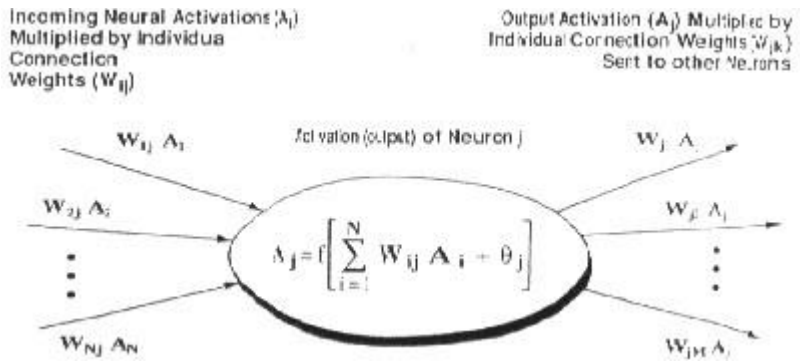
Gambar 8.12 Perceptron

Pada tahun 1969 Minsky dan Papert menulis sebuah buku di mana mereka menggambarkan keterbatasan Perceptrons lapisan tunggal. Dampak yang ditimbulkan buku itu luar biasa dan menyebabkan banyak peneliti jaringan saraf kehilangan minat mereka. Buku itu ditulis dengan sangat baik dan menunjukkan secara matematis bahwa perceptron lapisan tunggal tidak dapat melakukan beberapa operasi pengenalan pola dasar seperti menentukan paritas suatu bentuk atau menentukan apakah suatu bentuk terhubung atau tidak. Apa yang tidak mereka sadari, sampai tahun 80-an, adalah bahwa dengan pelatihan yang sesuai, perceptron bertingkat dapat melakukan operasi ini.

8.10 Proses Pembelajaran

Penghafalan pola dan respons jaringan selanjutnya dapat dikategorikan ke dalam dua paradigma umum:
pemetaan asosiatif di mana jaringan belajar untuk menghasilkan pola tertentu pada set unit input setiap kali pola tertentu lainnya diterapkan pada set unit input. Pemetaan asosiatif umumnya dapat dipecah menjadi dua mekanisme:
auto-association: pola input dikaitkan dengan dirinya sendiri dan status unit input dan output bertepatan. Ini digunakan untuk memberikan penyelesaian pola, yaitu untuk menghasilkan pola setiap kali sebagian atau pola terdistorsi disajikan. Dalam kasus kedua, jaringan sebenarnya menyimpan pasangan pola yang membangun hubungan antara dua set pola.
hetero-asosiasi: terkait dengan dua mekanisme penarikan:

penarikan terdekat-tetangga, di mana pola output yang dihasilkan sesuai dengan pola input yang disimpan, yang paling dekat dengan pola yang disajikan, dan interpolative recall, di mana pola keluaran adalah interpolasi yang bergantung pada kesamaan dari pola yang disimpan sesuai dengan pola yang disajikan. Paradigma lain, yang merupakan pemetaan asosiatif varian adalah klasifikasi, yaitu ketika ada satu set kategori tetap di mana pola input harus diklasifikasikan.
deteksi keteraturan di mana unit belajar untuk menanggapi sifat tertentu dari pola input. Sedangkan dalam pemetaan asosiatif jaringan menyimpan hubungan antar pola, dalam deteksi keteraturan respon dari setiap unit memiliki 'makna' tertentu. Jenis mekanisme pembelajaran ini penting untuk penemuan fitur dan representasi pengetahuan.
Setiap jaringan saraf memiliki pengetahuan yang terkandung dalam nilai bobot koneksi. Memodifikasi pengetahuan yang disimpan dalam jaringan sebagai fungsi pengalaman menyiratkan aturan pembelajaran untuk mengubah nilai bobot.



Gambar 8.13 Proses Pembelajaran

Informasi disimpan dalam matriks bobot W dari jaringan saraf. Belajar adalah penentuan bobot. Mengikuti cara pembelajaran dilakukan, kita dapat membedakan dua kategori utama jaringan saraf:
jaringan tetap yang bobotnya tidak dapat diubah, yaitu dW/dt=0. Dalam jaringan seperti itu, bobot ditetapkan secara apriori sesuai dengan masalah yang harus dipecahkan. jaringan adaptif yang mampu mengubah bobotnya, yaitu dW/dt not= 0. Semua metode pembelajaran yang digunakan untuk jaringan saraf adaptif dapat diklasifikasikan ke dalam dua kategori utama:
Pembelajaran terawasi yang menggabungkan guru eksternal, sehingga setiap unit keluaran diberi tahu apa yang seharusnya menjadi respons yang diinginkan terhadap sinyal masukan. Selama proses pembelajaran informasi global mungkin diperlukan. Paradigma pembelajaran terawasi meliputi pembelajaran koreksi kesalahan, pembelajaran penguatan dan pembelajaran stokastik.
Isu penting mengenai pembelajaran terawasi adalah masalah konvergensi kesalahan, yaitu meminimalkan kesalahan antara nilai unit yang diinginkan dan dihitung. Tujuannya adalah untuk menentukan satu set bobot yang meminimalkan kesalahan. Salah satu metode terkenal, yang umum untuk banyak paradigma pembelajaran adalah konvergensi kuadrat terkecil (LMS).
Pembelajaran tanpa pengawasan tidak menggunakan guru eksternal dan hanya didasarkan pada informasi lokal. Ini juga disebut sebagai pengorganisasian mandiri, dalam arti ia mengatur sendiri data yang disajikan ke jaringan dan mendeteksi properti kolektif mereka yang muncul. Paradigma pembelajaran tanpa pengawasan adalah pembelajaran bahasa Ibrani dan pembelajaran kompetitif.
Dari Saraf Manusia ke Saraf Buatan Aspek pembelajaran menyangkut perbedaan atau tidak dari fase terpisah, di mana jaringan dilatih, dan fase operasi berikutnya. Kami mengatakan bahwa jaringan saraf belajar secara off-line jika fase pembelajaran dan fase operasi berbeda. Sebuah jaringan saraf belajar on-line jika belajar dan beroperasi pada waktu yang sama. Biasanya, pembelajaran terawasi dilakukan secara offline, sedangkan pembelajaran tanpa pengawasan dilakukan secara online.

8.11 Fungsi Transfer

Perilaku JST (Artificial Neural Network) tergantung pada bobot dan fungsi input-output (fungsi transfer) yang ditentukan untuk unit. Fungsi ini biasanya jatuh ke dalam salah satu dari tiga kategori:

- linier (atau jalan)
- ambang
- sigmoid

Untuk unit linier, aktivitas output sebanding dengan total output tertimbang.
Untuk unit ambang batas, output diatur pada salah satu dari dua level, hal ini tergantung pada total input apakah lebih besar atau kurang dari beberapa nilai yang diinginkan yaitu nilai ambang batas.
Untuk unit sigmoid, ketika input berubah apkah output bervariasi terus menerus tetapi tidak linier. Unit sigmoid memiliki kemiripan yang lebih besar dengan neuron nyata daripada unit linier atau ambang batas, tetapi ketiganya harus dianggap sebagai perkiraan kasar.
Untuk membuat jaringan saraf yang melakukan beberapa tugas tertentu, kita harus memilih bagaimana unit terhubung satu sama lain (lihat gambar 4.1), dan kita harus mengatur bobot pada koneksi dengan tepat. Apakah mungkin koneksi dapat menentukan satu unit dapat mempengaruhi unit yang lain. Bobot menentukan kekuatan pengaruh.
Untuk mengajarkan jaringan tiga lapis untuk melakukan tugas tertentu dengan menggunakan prosedur berikut:

1. fungsi transfer dapat menyajikan jaringan untuk contoh pelatihan, terdiri dari pola aktifitas untuk input bersama dengan pola aktifitas untuk output.
2. Fungsi transfer menentukan kedekatan output aktual jaringan sehingga sama dengan output yang diinginkan.
3. Fungsi transfer memodifikasi koneksi untuk menghasilkan perkiraan jaringan yang lebih baik dari output yang diinginkan.

Contoh untuk mengilustrasikan prosedur pengajaran di atas:
Asumsikan bahwa kita ingin jaringan mengenali digit tulisan tangan. Kita mungkin menggunakan array, katakanlah, 256 sensor, masing-masing merekam ada atau tidaknya tinta di area kecil satu digit. Oleh karena itu jaringan akan membutuhkan 256 unit input untuk setiap sensor, 10 unit output untuk setiap jenis digit dan sejumlah hidden layer.
Proses yang direkam oleh sensor pada setiap jenis digit, jaringan harus memperoleh aktivitas tinggi di unit keluaran yang sesuai dan aktivitas rendah pada unit keluaran lainnya.
Untuk melatih jaringan, kami menyajikan gambar digit dan membandingkan aktivitas sebenarnya dari 10 unit keluaran dengan aktivitas yang diinginkan. Kami kemudian menghitung kesalahan, yang didefinisikan sebagai kuadrat dari perbedaan antara aktivitas yang sebenarnya dan yang diinginkan. Selanjutnya kita ubah bobot masing-masing koneksi sehingga dapat mengurangi kesalahan. Kami mengulangi proses pelatihan ini untuk banyak gambar yang berbeda dari setiap gambar yang berbeda dari setiap jenis digit sampai jaringan mengklasifikasikan setiap gambar dengan benar.
Untuk menerapkan prosedur ini kita perlu menghitung turunan kesalahan untuk bobot (EW) untuk mengubah bobot dengan jumlah yang sebanding dengan tingkat di mana kesalahan berubah saat bobot diubah. Salah satu cara bagaimana menghitung bobot adalah dengan sedikit mengganggu bobot dan mengamati bagaimana kesalahannya berubah. Tetapi metode itu tidak efisien karena memerlukan gangguan terpisah untuk masing-masing dari banyak bobot.
Cara lain bagaimana menghitung nilai bobot yaitu dapat menggunakan metode Back-propagation yang dijelaskan di bawah ini, dan saat ini telah menjadi salah satu alat terpenting untuk melatih jaringan saraf. Cara ini dikembangkan dengan independen oleh dua tim, satu dari tim Le Cun, Fogelman-Soulie dan Gallinari dari Prancis, yang lain adalah Williams, Rumelhart dan Hinton dari AS.

8.12. Aplikasi jaringan saraf

Mengingat deskripsi jaringan saraf ini dan cara kerjanya, aplikasi dunia nyata apa yang cocok untuk mereka? Jaringan saraf memiliki penerapan yang luas untuk masalah bisnis dunia nyata. Bahkan, mereka telah berhasil diterapkan di banyak industri.

Karena jaringan saraf paling baik dalam mengidentifikasi pola atau tren dalam data, jaringan saraf sangat cocok untuk kebutuhan prediksi atau peramalan termasuk:

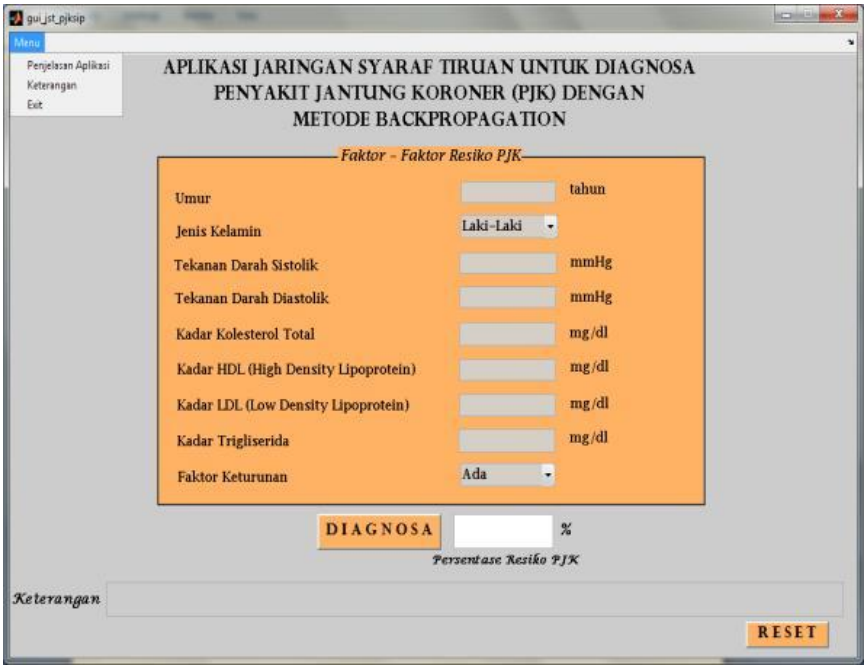
- perkiraan penjualan
- kontrol proses industri
- riset pelanggan
- validasi data
- manajemen risiko
- pemasaran sasaran

Tetapi untuk memberi Anda beberapa contoh yang lebih spesifik; JST juga digunakan dalam paradigma khusus berikut: pengenalan pembicara dalam komunikasi; diagnosis hepatitis; pemulihan telekomunikasi dari perangkat lunak yang rusak; interpretasi kata-kata Cina multimakna; deteksi ranjau bawah laut; analisis tekstur; pengenalan objek tiga dimensi; pengenalan kata tulisan tangan; dan pengenalan wajah.

Jaringan saraf dalam kedokteran

JST saat ini menjadi area penelitian yang lagi panas di bidang kedokteran dan diyakini bahwa mereka dalam beberapa tahun mendatang akan menerima aplikasi ekstensif untuk sistem biomedis. Waktu ini, penelitian lebih banyak dilakukan pada pemodelan bagian tubuh manusia dan pengenalan penyakit dari berbagai pemindaian (misalnya, kardiogram, pemindaian CAT, pemindaian ultrasonik, dll.).

Untuk mengenali penyakit menggunakan pemindaian tidak perlu meenggunkan algoritma khusus untuk mengidentifikasi penyakit apabila menggunakan Jaringan saraf sangat ideal. Jaringan saraf bekerja berdasarkan contoh sudah diproses pada pelatihan mengenali penyakit, sehingga tidak dibutuhkan rincian khusus. Tetapi yang diperlukan merupakan bagian kecil variasi penyakit untuk mewakili dari semua varisasi dari inputan. Sehingga ciri atau contoh yang sudah dipilih harus berkwalitas untuk menghasilkan sistem bekerja sesuai dengan yang diinginkan.



Gambar 8.14 Contoh Aplikasi jaringan syaraf tiruan untuk penyakit jantung koroner

Pemodelan dan Diagnosis Sistem Kardiovaskular

Neural Networks digunakan secara eksperimental untuk memodelkan sistem kardiovaskular manusia. Diagnosis dapat dicapai dengan membangun model sistem kardiovaskular individu dan membandingkannya dengan pengukuran fisiologis waktu nyata yang diambil dari pasien. Jika rutinitas ini dilakukan secara teratur, potensi kondisi medis yang berbahaya dapat dideteksi sejak dini dan dengan demikian membuat proses memerangi penyakit menjadi lebih mudah.

Model sistem kardiovaskular individu harus meniru hubungan antara variabel fisiologis (yaitu, denyut jantung, tekanan darah sistolik dan diastolik, dan laju pernapasan) pada tingkat aktivitas fisik yang berbeda. Jika suatu model disesuaikan dengan individu, maka model tersebut menjadi model kondisi fisik individu tersebut. Simulator harus dapat beradaptasi dengan

fitur dari setiap individu tanpa pengawasan seorang ahli. Ini panggilan untuk jaringan saraf.

Alasan lain yang membenarkan penggunaan teknologi JST, adalah kemampuan JST untuk memberikan sensor fusion yang merupakan penggabungan nilai dari beberapa sensor yang berbeda. Penggabungan sensor memungkinkan JST untuk mempelajari hubungan yang kompleks di antara nilai-nilai sensor individual, yang jika tidak, akan hilang jika nilai-nilai tersebut dianalisis secara individual. Dalam pemodelan dan diagnosis medis, ini menyiratkan bahwa meskipun setiap sensor dalam satu set mungkin hanya sensitif terhadap variabel fisiologis tertentu, JST mampu mendeteksi kondisi medis yang kompleks dengan menggabungkan data dari sensor biomedis individu.

Gambar 8.15 Contoh Aplikasi jaringan syaraf tiruan untuk Deteksi tulang

Elektronik Nose

JST digunakan secara eksperimental untuk mengimplementasikan hidung elektronik. Hidung elektronik memiliki beberapa aplikasi potensial dalam telemedicine. Telemedicine adalah praktik kedokteran jarak jauh melalui tautan komunikasi. Hidung elektronik akan mengidentifikasi bau di lingkungan bedah jarak jauh. Bau yang teridentifikasi ini kemudian akan ditransmisikan secara elektronik ke situs lain di mana sistem pembangkit pintu akan membuatnya kembali. Karena indra penciuman dapat menjadi indra yang penting bagi ahli bedah, telesmell akan meningkatkan operasi telepresent.

Gambar 8.16 Contoh Aplikasi jaringan syaraf tiruan untuk hidung elektronik

Dokter Instan

Sebuah aplikasi yang dikembangkan pada pertengahan 1980-an yang disebut "dokter instan" melatih jaringan saraf memori autoasosiatif untuk menyimpan sejumlah besar catatan medis, yang masing-masing mencakup informasi tentang gejala, diagnosis, dan perawatan untuk kasus tertentu. Setelah pelatihan, jaring dapat disajikan dengan input yang terdiri dari serangkaian gejala; kemudian akan menemukan pola tersimpan lengkap yang mewakili diagnosis dan pengobatan "terbaik".

Gambar 8.17 Contoh Aplikasi jaringan syaraf tiruan untuk Praktek Dokter

Jaringan Neural dalam bisnis

Jaringan saraf semakin banyak digunakan dalam aplikasi bisnis dunia nyata dan, dalam beberapa kasus, seperti deteksi penipuan, mereka telah menjadi metode pilihan. Penggunaannya untuk penilaian risiko juga berkembang dan telah digunakan untuk memvisualisasikan database yang kompleks untuk segmentasi pemasaran. Ledakan aplikasi ini mencakup berbagai kepentingan bisnis — mulai dari manajemen keuangan, melalui peramalan, hingga produksi. Kombinasi metode statistik, saraf, dan fuzzy sekarang memungkinkan studi kuantitatif langsung dilakukan tanpa memerlukan keahlian ilmu roket..

Gambar 8.18 Contoh Aplikasi jaringan syaraf tiruan untuk Peramalan Penjualan Mobil

Pemasaran

JST dapat diterapkan pada banyak masalah pengambilan keputusan pemasaran yang sebelumnya dapat diatasi dengan analisis statistik multivariat saja. Masalah-masalah tipikal berubah menjadi tugas segmentasi pasar dan lebih dominan pemodelan respons pasar, klasifikasi pola pengeluaran konsumen; analisis produk baru; identifikasi karakteristik pelanggan; perkiraan penjualan, pemasaran yang ditargetkan; dan pemodelan hubungan antara orientasi pasar dan kinerja Sebagian besar makalah yang dikutip secara eksplisit membandingkan pendekatan JST dengan metode tradisional termasuk terutama analisis diskriminan untuk tugas klasifikasi dan estimasi fungsi respons pasar dengan analisis regresi berganda. Poin terpenting untuk kegiatan riset di bidang pemasaran adalah kurangnya aplikasi pada data level individu. Masalah seperti ini ditemui dalam konteks pemodelan keputusan pembelian dan harus mengarah pada representasi JST dari perilaku pembelian dalam tradisi model stokastik perilaku konsumen [9]. Tabel menunjukkan beberapa penelitian terpilih yang telah menggunakan ANN di bidang pemasaran dan penjualan.

Gambar 8.19 Contoh Aplikasi jaringan syaraf tiruan untuk Prediksi suku bunga

Evaluasi Kredit

Risiko kredit keuangan adalah risiko kerugian keuangan yang timbul dari kemampuan atau ketidakmampuan pihak lawan untuk memenuhi kewajiban mereka yang disepakati dalam kontrak keuangan. Analisis risiko kredit adalah proses yang mengidentifikasi obligor dan mengkuantifikasi jumlah yang harus dibayar kembali pinjaman mereka dengan baik di muka. Analis mengadopsi salah satu atau kedua metode berikut untuk pemodelan risiko kredit:

- Penambangan data dan/atau pendekatan pembelajaran statistik
- Komputasi alami dan pemodelan matematika

Metrik utama dalam pemodelan risiko kredit adalah peringkat kredit (probabilitas default), eksposur saat default, dan kerugian yang diberikan default. Biasanya, peringkat kredit atau kemungkinan perhitungan default adalah masalah klasifikasi dan pohon regresi yang mengklasifikasikan pelanggan sebagai "berisiko" atau "tidak berisiko", atau memprediksi kelas berdasarkan data masa lalu. Meskipun analisis statistik tradisional dan model matematika banyak digunakan dalam berbagai skenario dalam analisis risiko kredit, model jaringan saraf lebih fleksibel dan mampu memodelkan fungsi non-linier yang kompleks daripada model statistik klasik seperti analisis diskriminan linier dan regresi logistik. Misalnya, model regresi logistik mudah diinterpretasikan karena kombinasi linier aditif dari input dan bobot, dan juga disesuaikan dengan algoritma pembelajaran. Tetapi akan memberikan hasil akurasi yang rendah pada hubungan non-linier yang kompleks. Namun, untuk model jaringan saraf yang menggunakan fungsi logistik, jumlah lapisan tersembunyi (H) yang lebih tinggi memungkinkannya untuk mempelajari hubungan non-linier yang kompleks. Bahkan, jaringan saraf dengan nilai H=0 setara dengan regresi logistik..

Gambar 8.20 Contoh Aplikasi jaringan syaraf tiruan untuk Pelatihan bank

8.13 Macam – Macam Algoritma

8.13.1 Algoritma Hebbian

Jaringan Hebb dikemukakan oleh Donald Hebb pada tahun 1949. Menurut aturan Hebb, bobot ditemukan meningkat secara proporsional dengan produk input dan output. Artinya dalam jaringan Hebb jika dua neuron saling berhubungan maka bobot yang terkait dengan neuron tersebut dapat ditingkatkan dengan perubahan celah sinaptik.

Jaringan ini cocok untuk data bipolar. Aturan belajar Hebbian umumnya diterapkan pada gerbang logika.

Bobot diperbarui sebagai:

W(new) = w(old) + x*y

Algoritma Pelatihan Untuk Aturan Pembelajaran Hebbian

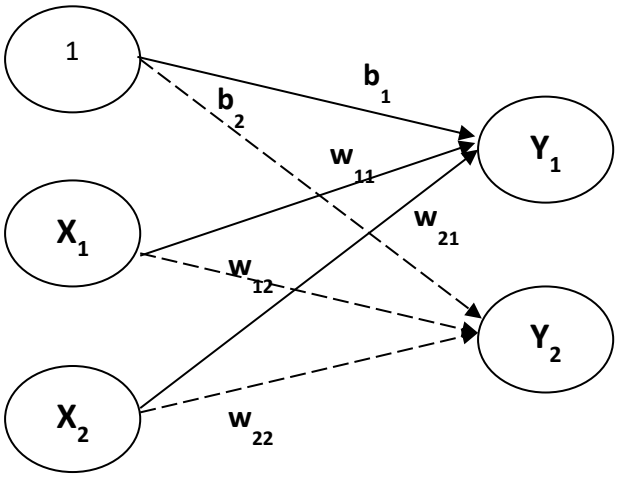
Langkah-langkah pelatihan algoritma adalah sebagai berikut:

Awalnya, bobot diatur ke nol, yaitu w =0 untuk semua input i =1 hingga n dan n adalah jumlah total neuron input.

Mari menjadi output. Fungsi aktivasi untuk input umumnya ditetapkan sebagai fungsi identitas.

Fungsi aktivasi untuk output juga diatur ke y= t.

Penyesuaian bobot dan bias disesuaikan ke:.



Gambar 8.21. Arsitektur jaringan Hebbian

- (w_i (new)) atau Nilai bobot yang baru diperoleh dari jumlah (w_i (old)) atau nilai bobot yang lama dengan (Δw) atau nilai perubahan bobot, sehingga bisa dirumuskan formula sbb :

  w_i (new) = w_i (old) + Δw_i

- Untuk nilai (Δw) diperoleh melalui :

  $\Delta w_i = x_i . y_i$

- dan untuk perubahan bias (Δb) adalah

  $\Delta b = y$

Apabila syarat berhentinya dipenuhi maka proses pembelajaran akan berhenti. Untuk menghentikan proses pembelajaran Jaringan Hebbian menggunakan separabilitas linier. Proses pembelajaran akan berjalan terus apabila separabilitas liniernya belum terpenuhi. Untuk proses pengenalan itu separabilitas linier masih digunakan.

Step 0: Inisialisasi semua bobot

$w_i = 0$ dumana i = 0 samapi ke n ;b = 0;

Step 1: Untuk setiap pasangan vektor (s & t), lakukan step 2 sampai 5.

Step 2: set nilai aktifasi unit input $x_i = s_i$

Step 3: set nilai aktifasi unit output $y_i = t_i$.

Step 4: Ubah nilai bobot dan bias

$w_i(new) = w_i(old) + x_i y_i$.

$b(new) = b(old) + y_i$.

Step 5: cek terhadap syarat stop menggunakan separabilitas linier.

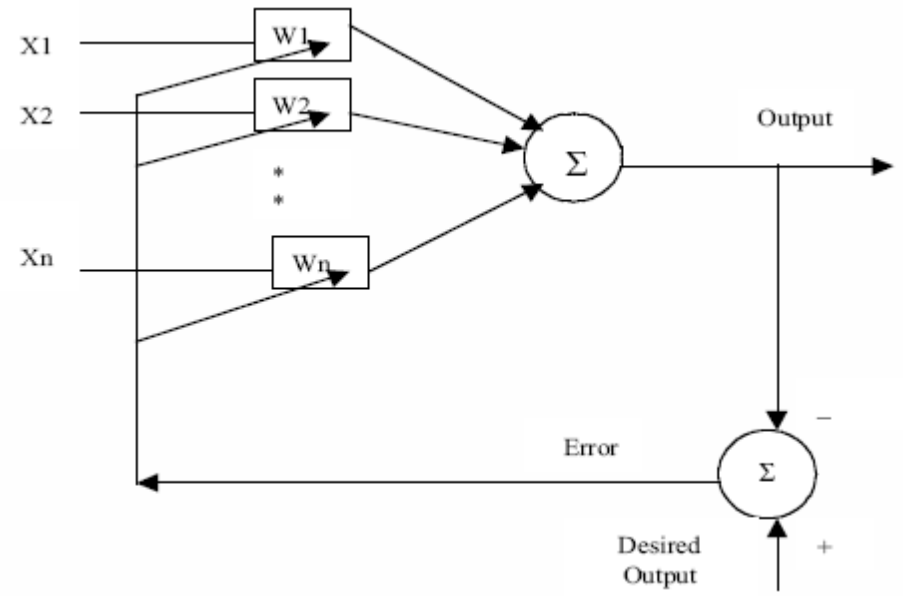
8.13.2 Algoritma Adaline

Sebuah ADAPtive LINEar Element (ADALINE) terdiri dari satu neuron tipe McCulloch-Pitts, dimana bobotnya ditentukan oleh hukum pelatihan kuadrat terkecil (LMS) ternormalisasi. Algoritma pembelajaran LMS pada awalnya diusulkan oleh Widrow dan Hoff. Aturan belajar ini juga disebut aturan delta. Ini adalah metode pelatihan yang diawasi dengan baik yang telah digunakan pada berbagai aplikasi

yang beragam. Perkiraan pemasangan kurva dapat juga dapat digunakan untuk melatih jaringan saraf . Tujuan pembelajaran pemasangan kurva adalah untuk menemukan permukaan yang paling cocok dengan data pelatihan. Pada bab selanjutnya akan dijelaskan implementasi algoritma LMS untuk backpropagation, dan algoritma curve fitting untuk jaringan fungsi radial basis.

Arsitektur ADALINE sederhana ditunjukkan pada Gambar 8.16. diamati bahwa struktur dasar ADALINE mirip dengan neuron linier dengan fungsi aktivasi f(.) menjadi satu linier dengan loop umpan balik ekstra. Karena ADALINE adalah perangkat linier, setiap kombinasi dari unit-unit ini dapat dicapai dengan menggunakan satu unit.

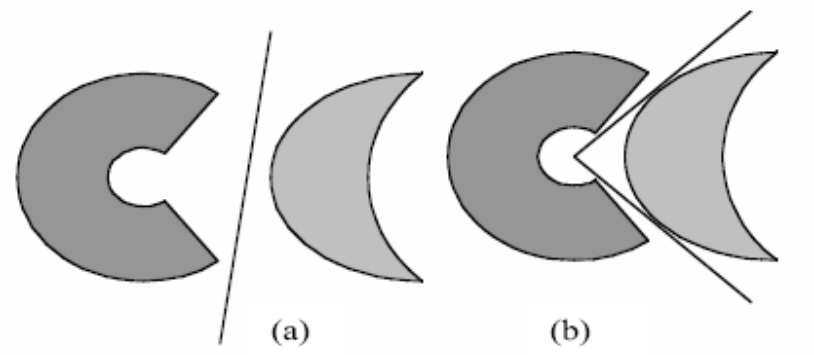
Selama fase pelatihan ADALINE, vektor input $X \in R^n$: $[X = x1\ x2\ x3\ .....\ x_n]^T$ serta output yang diinginkan disajikan ke jaringan. Bobot disesuaikan secara adaptif berdasarkan aturan delta. Setelah ADALINE dilatih, vektor input yang disajikan ke jaringan dengan bobot tetap akan menghasilkan output skalar. Oleh karena itu, jaringan melakukan pemetaan pemetaan n dimensi ke nilai skalar. Fungsi aktivasi tidak digunakan selama fase pelatihan. Setelah bobot disesuaikan dengan benar, respons unit yang dilatih dapat diuji dengan menerapkan berbagai input, yaitu: tidak dalam set pelatihan. Jika jaringan menghasilkan respons yang konsisten hingga tingkat tinggi dengan input pengujian, dikatakan bahwa jaringan dapat digeneralisasi. Oleh karena itu, proses pelatihan dan generalisasi adalah dua atribut penting dari jaringan.



Gambar 8.22. Arsitektur Adaline

Dalam praktiknya, ADALINE biasanya digunakan untuk membuat keputusan biner. Oleh karena itu, output dikirim melalui ambang biner. Realisasi beberapa gerbang logika seperti AND, NOT dan OR adalah aplikasi umum dari ADALINE. Hanya fungsi logika yang dapat dipisahkan secara linier yang dapat direalisasikan oleh ADALINE.

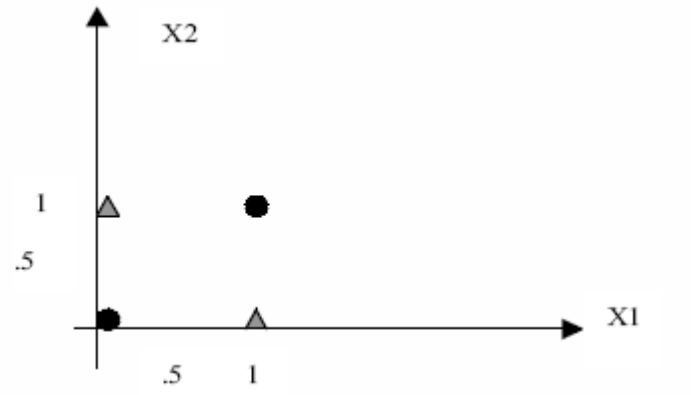
Agar ADALINE tunggal berfungsi dengan baik sebagai pengklasifikasi, pola input harus dapat dipisahkan secara linier. Ini menyiratkan bahwa pola yang akan diklasifikasikan harus cukup terpisah satu sama lain untuk memastikan permukaan keputusan terdiri dari hyperplane tunggal seperti garis lurus tunggal dalam ruang dua dimensi. Konsep ini diilustrasikan pada Gambar 8.17 untuk pola dua dimensi.



Gambar 8.23 : Sepasang Pola yang Dapat Dipisahkan Secara Linier (a), dan Pola yang Tidak Dapat Dipisahkan Secara Linier (b).

Contoh klasik dari pemetaan yang tidak dapat dipisahkan adalah fungsi gerbang XOR (eksklusif OR). Tabel 8.1 menunjukkan pola input-output dari masalah ini. Gambar 8.18 menunjukkan lokasi keluaran simbolis dari fungsi XOR yang sesuai dengan empat pola masukan pada bidang X1- X2. Tidak ada cara untuk menggambar garis lurus tunggal sehingga lingkaran berada di satu sisi garis dan tanda segitiga di sisi lain. Oleh karena itu, seorang ADALINE tidak dapat mewujudkan fungsi ini.
Tabel 8.1: Hubungan Input/Output untuk XOR.

X1	*X2*	*Output*
0	0	0
0	1	1
1	0	1
1	1	0



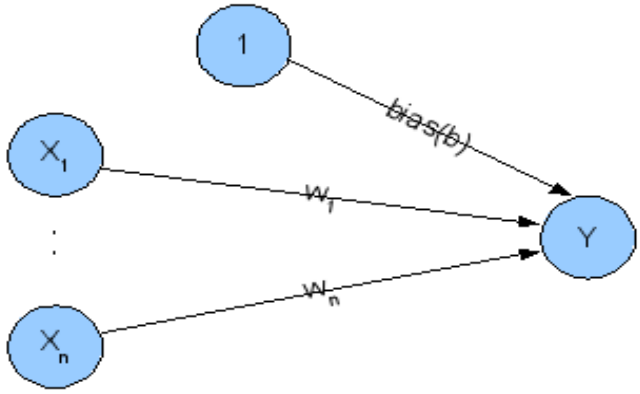
Gambar 8.24: Output XOR pada Bidang X1- X2.

Salah satu pendekatan untuk menyelesaikan masalah pemisahan nonlinier ini adalah dengan menggunakan jaringan MADALINE (Multiple ADALINE). Struktur dasar jaringan MADALINE terdiri dari menggabungkan beberapa ADALINE dengan fungsi aktivasi korespondensinya menjadi satu struktur maju. Ketika bobot yang sesuai dipilih, jaringan mampu mengimplementasikan pemetaan yang rumit dan tidak dapat dipisahkan seperti masalah gerbang XOR.

8.13.3 Algoritma Perceptron

Algoritma Pembelajaran Perceptron, awalnya diusulkan oleh Frank Rosenblatt pada tahun 1943, kemudian disempurnakan dan dianalisis dengan cermat oleh Minsky dan Papert pada tahun 1969. Ini

adalah posting lanjutan dari posting saya sebelumnya tentang model neuron McCulloch-Pitts dan model Perceptron.
Jaringan ini memiliki n input (x1 hingga xn) dengan masing-masing input memiliki bobot wn. Sebuah bias ditambahkan ke jaringan dengan learning rate = 1. Output yang dihasilkan adalah Y. Karena metode ini adalah pembelajaran terawasi ada ambang batas yang harus dilewati.



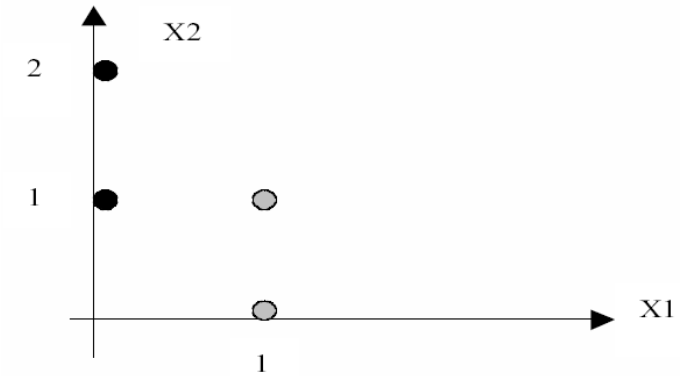
Gambar 8.25. Arsitektur Perceptron

Berikut beberapa notasi dan fungsi yang perlu kita ketahui:

1. 1. xi = input ke-i, b = bias, = bilangan pembelajaran (0<α≤1), wi = bobot setiap input,
2. ti = ambang batas input ke-I
3. 2. Output yang dihasilkan adalah: Yin = wi.xi + .b
4. 3. Sedangkan keluaran dalam bentuk biner (y) adalah 1 jika 1 Yin dan 0 jika Yin 0
5. 4. Perubahan nilai bobot (w) dan bias (b) selamat belajar dilambangkan sebagai:

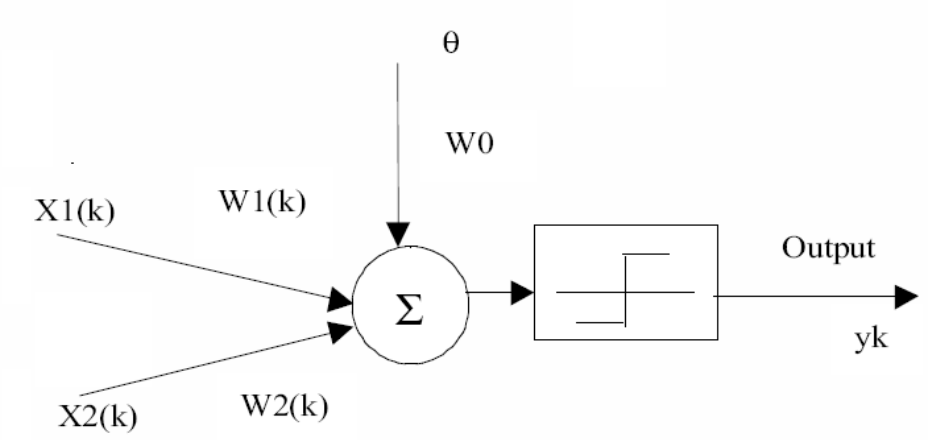
$W_{baru} = w_{lama} + x$	jika Y < t
$W_{baru} = w_{lama} - x$	jika Y > t
$W_{baru} = w_{lama}$	jika Y = t

Contoh :
Mari kita perhatikan kelas pola C1 dan C2, di mana C1: {(0,2), (0,1)} dan C2: {(1,0), (1,1)}. Tujuannya adalah untuk mendapatkan permukaan keputusan berdasarkan pembelajaran perceptron. Grafik 2-D untuk data di atas ditunjukkan pada Gambar 8.22



Gambar 8.26: Plot 2-D dari Kumpulan Data Input untuk Contoh

Penyelesaian :
Karena vektor input terdiri dari dua elemen , struktur perceptron adalah sebagai berikut:



Gambar 8.27 : Struktur Perceptron untuk Contoh

Untuk mempermudah, mari kita asumsikan η=1 dan vektor bobot awal W(1)=[0 0]. Bobot iterasinya adalah sebagai berikut:
Iterasi 1 :

$$W^T(1)\,.\,x(1) = \begin{bmatrix}0\ 0\end{bmatrix}\begin{bmatrix}0\\2\end{bmatrix} = 0$$

Weight Update:

$$W(2) = W(1) + x(1) = \begin{bmatrix}0\\0\end{bmatrix} + \begin{bmatrix}0\\2\end{bmatrix} = \begin{bmatrix}0\\2\end{bmatrix}$$

Iterasi 2 :

$$W^T(2).x(2) = \begin{bmatrix}0\ 2\end{bmatrix}\begin{bmatrix}0\\1\end{bmatrix} = 2 > 0$$

Weight Update:

W(3) = W(2)

Iterasi 3 :

$$W^T(3)\,.\,x(3) = \begin{bmatrix}0\ 2\end{bmatrix}\begin{bmatrix}1\\0\end{bmatrix} = 0$$

Weight Update:

$$W(4) = W(3) - x(3) = \begin{bmatrix}0\\2\end{bmatrix} - \begin{bmatrix}1\\0\end{bmatrix} = \begin{bmatrix}-1\\2\end{bmatrix}$$

Iterasi 4 :

$$W^T(4)\,.\,x(4) = \begin{bmatrix}-1\ 2\end{bmatrix}\begin{bmatrix}1\\1\end{bmatrix} = 1$$

Weight Update:

$$W(5) = W(4) - x(4) = \begin{bmatrix}-1\\2\end{bmatrix} - \begin{bmatrix}1\\1\end{bmatrix} = \begin{bmatrix}-2\\1\end{bmatrix}$$

Sekarang jika kita melanjutkan prosedur, perceptron mengklasifikasikan dua kelas dengan benar pada setiap instance. Misalnya untuk iterasi kelima dan keenam:

Iterasi 5 :

$$W^T(5).x(5) = [-2\ 1]\begin{bmatrix}0\\2\end{bmatrix} = 2 > 0$$ Klasifikasi sudah benar

Iterasi 6 :

$$W^T(6).x(6) = [-2\ 1]\begin{bmatrix}0\\2\end{bmatrix} = 1 > 0$$ Klasifikasi sudah benar

Dengan cara yang sama untuk iterasi ketujuh dan kedelapan, hasil klasifikasi memang benar.

Iterasi 7 :

$$W^T(7).x(7) = [-2\ 1]\begin{bmatrix}1\\0\end{bmatrix} = -2 < 0$$ Klasifikasi sudah benar

Iterasi 8 :

$$W^T(8).x(8) = [-2\ 1]\begin{bmatrix}1\\1\end{bmatrix} = -1 < 0$$ Klasifikasi sudah benar

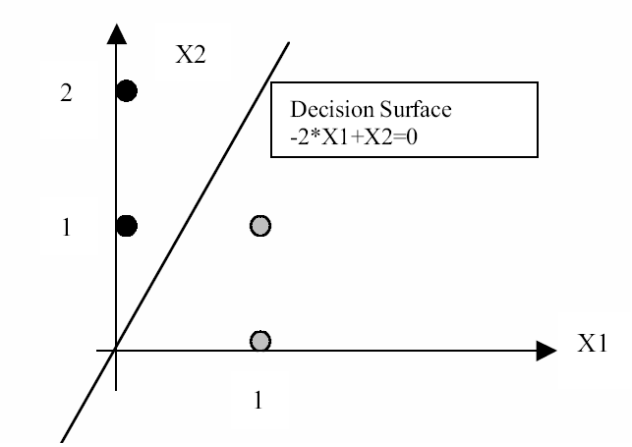
Oleh karena itu, algoritma konvergen dan permukaan keputusan untuk perceptron di atas adalah sebagai berikut:

$d(x) = -2X_1 + X_2 = 0$

Sekarang, mari kita perhatikan data input {1,2}, yang tidak ada dalam set pelatihan. Jika kita menghitung outputnya:

$$Y = W^T.x = [-2\ 1]\begin{bmatrix}2\\1\end{bmatrix} = -3 < 0$$

Output Y milik kelas C2 seperti yang diharapkan.



Gambar 8.28: Permukaan Keputusan untuk Contoh

Algoritma pembelajaran perceptron (Aturan Delta) dapat diringkas sebagai berikut:
Langkah 1: Inisialisasi bobot W1, W2…Wn dan ambang ke nilai acak kecil.
Langkah 2: Hadirkan input baru X1, X2,..Xn dan output yang diinginkan dk .
Langkah 3: Hitung output aktual berdasarkan rumus berikut:

$$y_k = f_h \left(\sum_{i=1}^{n} (X_i \ W_i) - \ \theta_k \right)$$

Langkah 4: Sesuaikan bobot sesuai dengan persamaan berikut:

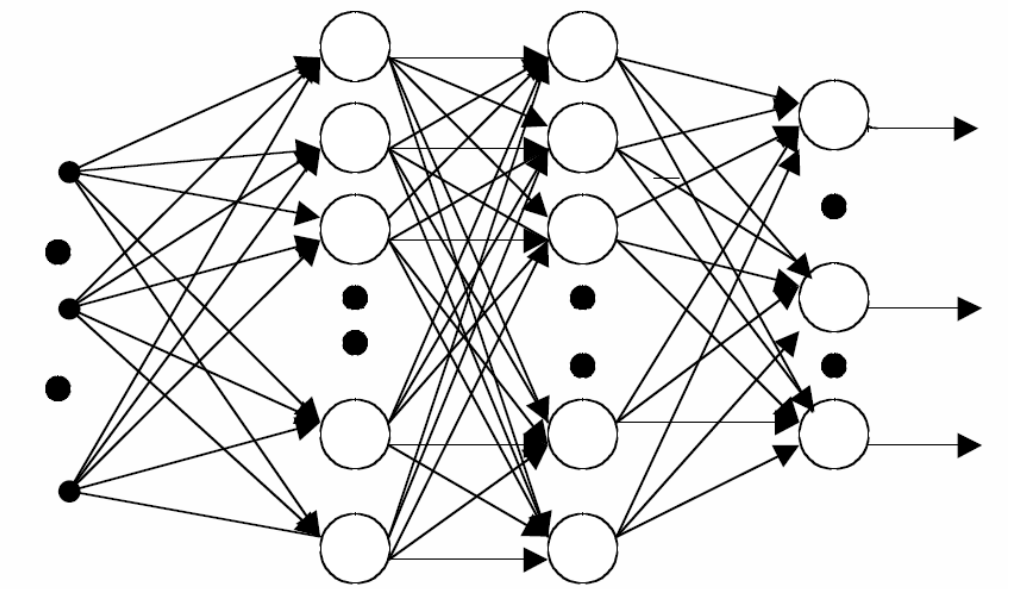
$$W_i(baru) = \ W_i(lama) + \ \eta \ (d_k - \ y_k)x_i, 0 \ \leq i \ \leq N$$

Dimana adalah fraksi gain positif kurang dari 1 dan dk adalah output yang diinginkan. Perhatikan bahwa bobot tetap sama jika jaringan membuat keputusan yang benar.
Langkah 5: Ulangi prosedur pada langkah 2−4 sampai tugas klasifikasi selesai.

Mirip dengan ADALINE, jika pola input yang disajikan dapat dipisahkan secara linier, maka algoritma perceptron di atas akan melakukan konvergensi dan memposisikan hyperplane keputusan di antara dua kelas yang terpisah. Di sisi lain, jika input tidak dapat dipisahkan dan distribusinya tumpang tindih, maka batas keputusan dapat berosilasi terus menerus. Modifikasi prosedur konvergensi perceptron adalah pemanfaatan Least Mean Square (LMS) dalam hal ini. Algoritma yang membentuk solusi LMS disebut juga dengan Widrow-Hoff. Algoritme LMS mirip dengan prosedur di atas kecuali nonlinier logika ambang batas, menggantikan nonlinieritas terbatas yang keras. Bobot dikoreksi pada setiap jejak dengan jumlah yang bergantung pada perbedaan antara nilai yang diinginkan dan nilai aktual. Berbeda dengan pembelajaran di ADALINE, aturan pembelajaran perceptron telah terbukti mampu memisahkan rangkaian pola pelatihan yang dapat dipisahkan secara linier.

8.13.4 Algoritma Multi-Layer Perceptron

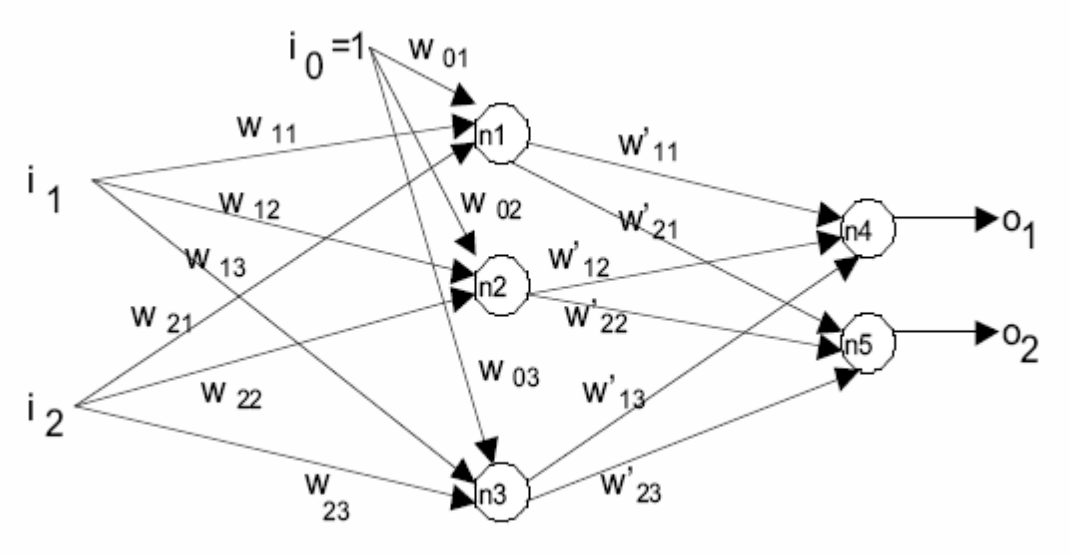
Perceptron multi-layer mewakili generalisasi dari single-layer perceptron seperti yang dijelaskan pada bagian sebelumnya. Perceptron lapisan tunggal membentuk wilayah keputusan setengah bidang. Di sisi lain perceptron multi-layer dapat membentuk wilayah keputusan yang kompleks dan dapat memisahkan berbagai pola input. Kemampuan multi-layer perceptron berasal dari non-linearitas yang digunakan dalam node. Jika node adalah elemen linier, maka jaringan lapisan tunggal dengan bobot yang sesuai dapat digunakan sebagai pengganti perceptron dua atau tiga lapis. Gambar 2.17 menunjukkan struktur jaringan saraf perceptron multi-layer yang khas. Seperti yang diamati itu terdiri dari lapisan-lapisan berikut:



Gambar 8.29: Perceptron multi-lapisan.

Input Layer: Lapisan neuron yang menerima informasi dari sumber eksternal, dan meneruskan informasi ini ke jaringan untuk diproses. Ini mungkin berupa input sensorik atau sinyal dari sistem lain di luar yang dimodelkan.
Hidden Layer: Lapisan neuron yang menerima informasi dari lapisan input dan memprosesnya secara tersembunyi. Ia tidak memiliki koneksi langsung ke dunia luar (input atau output). Semua koneksi dari lapisan tersembunyi ke lapisan lain dalam sistem.
Output Layer: Lapisan neuron yang menerima informasi yang diproses dan mengirimkan sinyal output keluar dari sistem.
Bias: Bekerja pada neuron seperti offset. Fungsi bias adalah untuk memberikan ambang batas untuk aktivasi neuron. Input bias terhubung ke masing-masing neuron tersembunyi dan output dalam jaringan.

Pemetaan Input-Output
Pemetaan input/output jaringan dibuat berdasarkan bobot dan fungsi aktivasi neuron pada lapisan input, hidden, dan output. Jumlah neuron input sesuai dengan jumlah variabel input dalam jaringan buatan, dan jumlah neuron output sama dengan jumlah variabel output yang diinginkan. Jumlah neuron di lapisan tersembunyi tergantung pada aplikasi NN tertentu. Sebagai contoh, perhatikan jaringan feed-forward dua lapis berikut dengan tiga neuron di hidden layer dan dua neuron di layer kedua:



Gambar 8.30: Contoh Multi-layer Perceptron.

Seperti yang ditunjukkan, input terhubung ke setiap neuron di lapisan tersembunyi melalui bobot yang sesuai. Sebuah bobot nol dalam menunjukkan tidak ada koneksi. Misalnya, jika W23 = 0, tersirat bahwa tidak ada koneksi antara input kedua (i2) dan neuron ketiga (n3) . Output dari lapisan terakhir dianggap sebagai output dari jaringan.
Struktur setiap neuron dalam satu lapisan mirip dengan arsitektur seperti yang dijelaskan pada bagian 2.5. Meskipun fungsi aktivasi untuk satu neuron dapat berbeda dari neuron lain dalam suatu lapisan , untuk kesederhanaan struktural , neuron serupa biasanya dipilih dengan dalam lapisan. Kumpulan data input (atau informasi sensorik) disajikan ke lapisan input. Lapisan ini terhubung ke lapisan tersembunyi pertama. Jika ada lebih dari satu lapisan tersembunyi, lapisan tersembunyi terakhir harus terhubung ke lapisan keluaran jaringan. Pada fase pertama, kita akan memiliki hubungan linier berikut untuk setiap lapisan:

$A_1 = W_1 X$

di mana A1 adalah vektor kolom yang terdiri dari m elemen, W1 adalah matriks bobot m×n dan X adalah vektor input kolom berdimensi n. Untuk contoh di atas, tingkat aktivitas linier dari lapisan tersembunyi (neuron n1 hingga n3) dapat dihitung sebagai berikut:

$$\begin{cases} a_{11} = w_{11} i_1 + w_{21} i_2 \\ a_{12} = w_{12} i_1 + w_{22} i_2 \\ a_{13} = w_{13} i_1 + w_{23} i_2 \end{cases}$$

Vektor keluaran untuk lapisan tersembunyi dapat dihitung dengan rumus berikut:

O1 = F .A1

dimana A1 didefinisikan dalam Persamaan n2 .12, dan O1 adalah vektor kolom keluaran dari lapisan tersembunyi dengan elemen m. F adalah matriks diagonal yang terdiri dari fungsi aktivasi non-linier sehingga f lapisan tersembunyi pertama :

$$F = \begin{bmatrix} f_1(.) & 0 & 0 & ... & 0 \\ 0 & f_2(.) & & & 0 \\ . & & .. & & .. \\ . & & & .. & 0 \\ 0 & 0 & ... & 0 & f_m(.) \end{bmatrix}$$

Sebagai contoh, jika semua fungsi aktivasi untuk neuron pada lapisan tersembunyi pada Gambar 2.26 dipilih dengan cara yang sama , maka output dari neuron n1 hingga n3 dapat dihitung sebagai berikut:

$$\begin{cases} O_{11} = f(a_{11}) \\ O_{12} = f(a_{12}) \\ O_{13} = f(a_{13}) \end{cases}$$

Dengan cara yang sama, output dari lapisan tersembunyi lainnya dapat dihitung. Output dari jaringan dengan hanya satu lapisan tersembunyi menurut Persamaan 2.14 adalah sebagai berikut:

$$\begin{aligned} A_2 &= W_2 . O_1 \\ O_2 &= G . A_2 \end{aligned}$$

Dimana A2 adalah vektor tingkat aktivitas lapisan keluaran dan O2 adalah keluaran q dari jaringan. G adalah matriks diagonal yang terdiri dari fungsi aktivasi non linier dari lapisan output:

$$G = \begin{bmatrix} g_1(.) & 0 & 0 & ... & 0 \\ 0 & g_2(.) & & & 0 \\ . & & .. & & .. \\ . & & & .. & 0 \\ 0 & 0 & ... & 0 & g_q(.) \end{bmatrix}$$

Untuk Gambar 2.18, tingkat aktivitas neuron keluaran n4 dan n5 dapat dihitung sebagai berikut:

$$\begin{cases} a_{21} = W'_{11} O_{11} + W'_{12} O_{21} + W'_{13} O_{31} \\ a_{22} = W'_{21} O_{11} + W'_{22} O_{21} + W'_{23} O_{31} \end{cases}$$

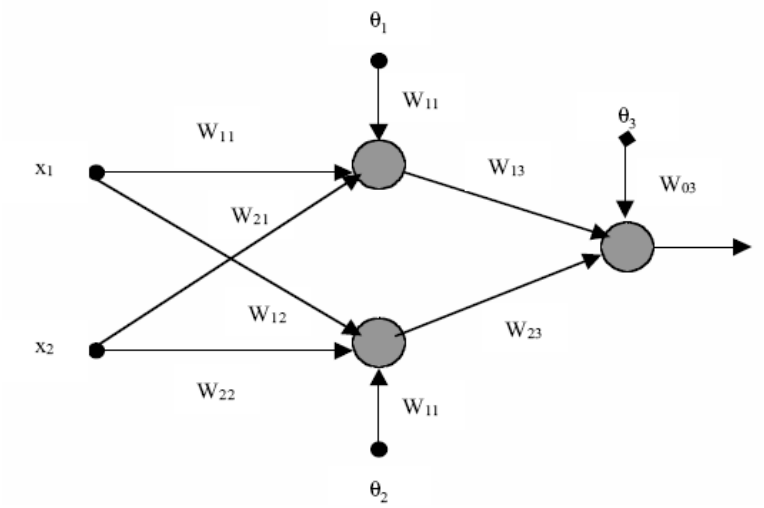
Dua keluaran jaringan dengan fungsi aktivasi yang sama dapat dihitung sebagai berikut:

$$\begin{cases} O_1 = g(a_{21}) \\ O_2 = g(a_{22}) \end{cases}$$

Oleh karena itu, pemetaan input-output dari perceptron multi-layer dibuat menurut hubungan 2.12−2.22. Selanjutnya, output dari jaringan dapat dihitung menggunakan pemetaan nonlinier tersebut dan set data input.

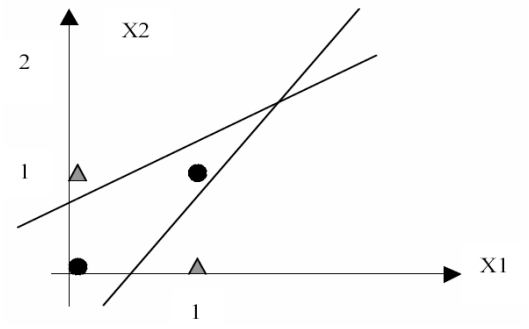
Realisasi XOR

Seperti yang ditunjukkan pada bagian 2.4, perceptron lapisan tunggal tidak dapat mengklasifikasikan pola input yang tidak dapat dipisahkan secara linier seperti gerbang Exclusive OR (XOR). Masalah ini dapat dianggap sebagai kasus khusus dari masalah pemetaan nonlinier yang lebih umum. Dalam masalah XOR, kita perlu mempertimbangkan empat sudut persegi satuan yang sesuai dengan pola input. Kami dapat memecahkan masalah dengan perceptron multi-layer dengan satu hidden layer seperti terlihat pada Gambar 8.31.



Gambar 8.31: Arsitektur Neural Network untuk Memecahkan Masalah XOR.

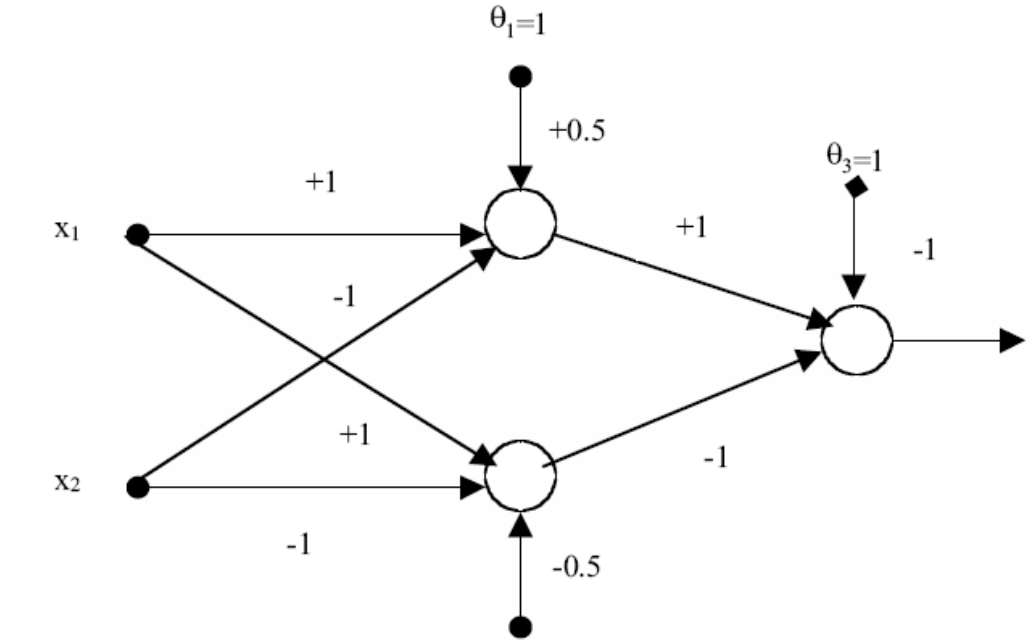
Dalam konfigurasi di atas, model McCulloh-Pitts mewakili setiap neuron, yang menggunakan fungsi aktivasi batas keras. Dengan pemilihan bobot jaringan yang tepat, XOR dapat diimplementasikan menggunakan permukaan keputusan seperti yang ditunjukkan pada Gambar 2.28.



Gambar 8.32: Permukaan Keputusan untuk Memecahkan Masalah XOR.

Contoh :

Misalkan bobot dan bias dipilih seperti yang ditunjukkan pada Gambar 2.29. Model McCulloh-Pitts mewakili setiap neuron (fungsi aktivasi batas keras biner). Tunjukkan bahwa jaringan memecahkan masalah XOR. Selain itu, gambarkan batasan keputusan yang dibangun oleh jaringan.

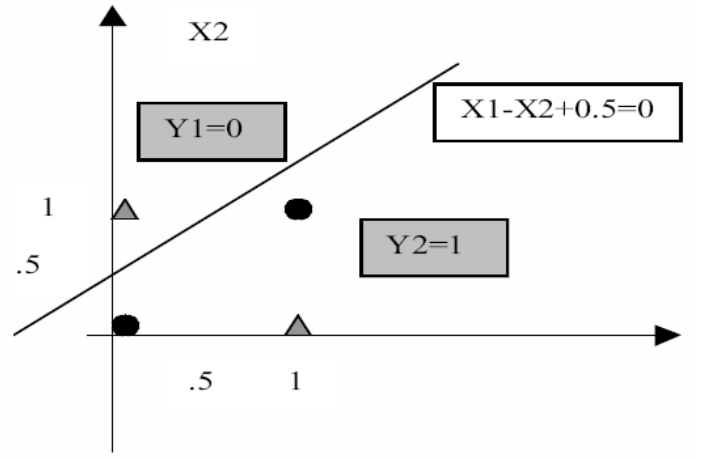


Gambar 8.33: Arsitektur Jaringan Syaraf Sebagai Contoh

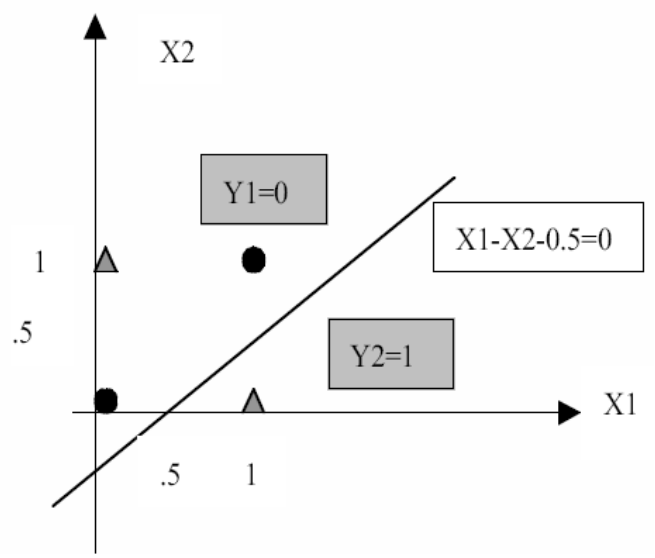
Pada Gambar 2.29, misalkan keluaran neuron (sebelum fungsi aktivasi) dinotasikan sebagai O1, O2, dan O3. Output dari penjumlahan poin pada layer pertama adalah sebagai berikut:

$$O_1 = x_1 - x_2 + 0.5$$
$$O_2 = x_1 - x_2 - 0.5$$

Dengan fungsi biner terbatas keras, output y1 dan y 2 ditunjukkan pada Gambar 2.30 dan 2.31.



Gambar 8.34: Permukaan Keputusan untuk Neuron 1 dari Contoh

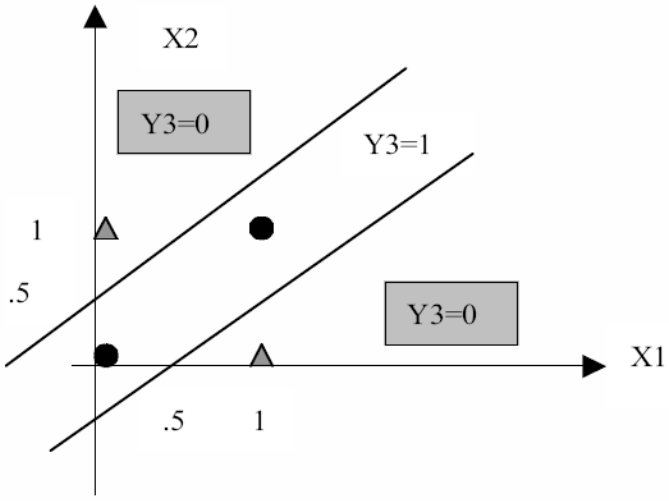


Gambar 8.35: Permukaan Keputusan untuk Neuron 2 dari Contoh

Output dari penjumlahan poin pada lapisan kedua adalah:

$O_3 = y_1 - y_2 - 1$

Batas keputusan jaringan ditunjukkan pada Gambar 2.32. Oleh karena itu, realisasi XOR dapat dilakukan dengan pemilihan bobot yang sesuai menggunakan Gambar 2.27.



Gambar 8.36: Permukaan Keputusan untuk Contoh

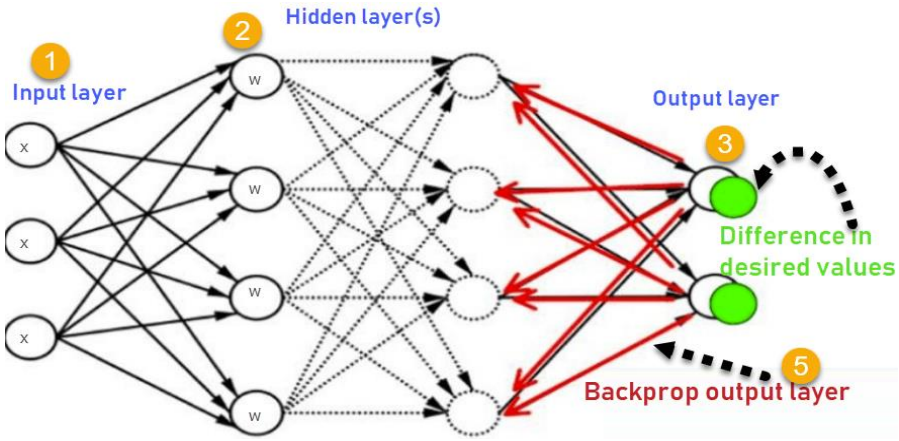
8.13.5 Algoritma Backpropagation

Backpropagation (BP) merupakan JST multi-layer. Penemuannya mengatasi kelemahan JST dengan layer tunggal yang mengakibatkan perkembangan JST sempat tersendat disekitar tahun 1970. Algoritma BP merupakan generalisasi aturan delta (Widrow-Hoff), yaitu menerapkan metode *gradient descent* untuk meminimalkan error kuadrat total dari keluaran yang dihitung oleh jaringan.

− Banyak aplikasi yang dapat diselesaikan dengan BP, akibatnya JST semakin banyak diminati orang.

− JST layer tunggal memiliki kelemahan dalam pengenal pola. Hal ini di atasi dengan menambah satu/beberapa layer tersembunyi diantara layar masukan dan layer keluaran. Banyak layar tersembunyi memiliki kelebihan manfaat untuk beberapa kasus, namun pelatihannya memerlukan waktu yang lama.
− Sesuai dengan ide dasar JST, BP melatih jaringan untuk memperoleh keseimbangan antara "kemampuan jaringan" untuk mengenali pola yang digunakan selama pelatihan dan "kemampuan jaringan" merespon secara benar terhadap pola masukan yang serupa (tapi tidak sama) dengan pola pelatihan.

Arsitektur BP



Gambar 8.37: Arsitektur BackPropagation.

Keuntungan paling menonjol dari Backpropagation adalah:

- Backpropagation cepat, sederhana dan mudah diprogram
- Itu tidak memiliki parameter untuk disetel selain dari jumlah input
- Ini adalah metode yang fleksibel karena tidak memerlukan pengetahuan sebelumnya tentang jaringan
- Ini adalah metode standar yang umumnya bekerja dengan baik
- Tidak perlu disebutkan secara khusus fitur fungsi yang akan dipelajari.

Algoritma Pelatihan

Pelatihan BP mencakup 3 fase: i) fase umpan maju dari pola pelatihan input. Pola input maju dari lapisan input ke lapisan output dengan fungsi aktivasi yang ditentukan; ii) fase backpropagation dari kesalahan terkait. Selisih antara keluaran dan target merupakan kesalahan yang terjadi. Penjelasannya disebarkan mundur, dimulai dari garis yang berhubungan langsung dengan unit-unit pada layar keluaran; iii) fase modifikasi berat.

i) Fase propagasi maju

Propagasi maju, sinyal input (x1) disebarkan ke lapisan tersembunyi menggunakan fungsi aktivasi yang ditentukan. Keluaran dari unit tersembunyi (Zj) kemudian dipropagasi maju lagi ke lapisan tersembunyi berikutnya dengan fungsi aktivasi yang telah ditentukan. Begitu seterusnya hingga menghasilkan keluaran jaringan (yk).

Selanjutnya keluaran jaringan (yk) dibandingkan dengan target yang ingin dicapai (tk). Perbedaan tk – yk adalah error yang terjadi. Jika kesalahan ini kurang dari batas toleransi yang ditentukan, maka iterasi dihentikan. Jika kesalahan masih lebih besar dari batas toleransi, maka setiap jalur jaringan akan mengurangi kesalahan

ii) Fase Propagasi Mundur

Berdasarkan kesalahan tk – yk, dihitung faktor k (k= 1, ..., m) yang digunakan untuk mendistribusikan kesalahan dalam satuan Yk ke semua satuan tersembunyi yang terhubung langsung ke Yk. k juga digunakan untuk mengubah bobot garis yang berhubungan langsung dengan unit keluaran.

Dengan cara yang sama, faktor j dihitung pada setiap lapisan tersembunyi sebagai dasar untuk perubahan bobot semua garis yang berasal dari unit tersembunyi pada lapisan di bawahnya. Begitu seterusnya hingga semua faktor di unit tersembunyi terhubung langsung ke unit input yang dihitung.
iii) Tahap Modifikasi Berat
Setelah semua faktor dihitung, bobot semua garis akan diputar secara bersamaan. Perubahan bobot garis berdasarkan faktor neuron di lapisan atas.

Misalnya, mengubah bobot garis menuju ke lapisan berdasarkan k di unit keluaran
Ketiga fase tersebut diulangi sampai kondisi terpenuhi. Umumnya kondisi yang sering digunakan adalah jumlah iterasi atau error. Iterasi akan dihentikan jika jumlah iterasi yang dilakukan telah melebihi jumlah maksimum iterasi yang ditentukan, atau jika error yang terjadi kurang dari batas toleransi yang ditentukan.
Algoritma BP dengan satu lapisan tersembunyi dengan fungsi aktivasi Sigmoid biner adalah sebagai berikut:
1. Inisialisasi semua bobot dengan angka acak kecil
2. Jika kondisi berhenti tidak terpenuhi, lakukan langkah 2 – 9
3. Untuk setiap pasangan data latih, lakukan langkah 3 – 8
Propagasi Maju Fase I
4. Setiap unit menerima sinyal dan memonitor ke unit tersembunyi di atasnya
. menghitung semua output dalam unit tersembunyi Zj(j = 1....p)
$z_{\_net\ j} = v_{0j} + \Sigma = pjijivx1$
$z_j = f(z_{\_net\ j}) = j\_net\_ze{-}{+}11$
5. Hitung semua keluaran jaringan di unit Y_k (k= 1....m)
$y_{\_net\ k} = w_{0j} + \Sigma = mkjkjwz1$
$y_k = f(y_{\_net\ k}) = k\_net\_ye{-}{+}11$
Propagasi Terbalik Fase II
6. Hitung faktor unit output berdasarkan error pada setiap unit output Yk (k= 1,2, ..., m)
k = (tk – yk) f' (y_net k) = (tk – yk) yk (1 – yk)
k adalah kesalahan yang akan digunakan dalam mengubah berat lapisan di bawahnya (langkah 7)
hitung laju perubahan bobot wjk (yang nantinya akan digunakan untuk mengubah bobot wjk) dengan learning rate
wjk = k zj (k= 1,2, ..., m ; j = 0,1,2, ..., p)
7. Hitung faktor satuan tersembunyi berdasarkan kesalahan pada setiap satuan tersembunyi Zj (j = 1....p)
_net j = =mkjkkw1δ
Faktor unit tersembunyi
j = _net j f'(z_net j) = _net j zj (1 – zj)
hitung tingkat perubahan bobot vij (yang akan digunakan untuk mengubah vij)
vij = j xi (j = 1,2, ..., p ; i = 0,1,2, ..., n)
Modifikasi Berat Tahap III
8. Hitung semua perubahan berat badan
Perubahan berat baris ke unit output:
wjk (baru) = wjk (lama) + wjk (k=1,2, ...m ; j =0,1,2,...,p)
Ubah bobot garis yang mengarah ke unit tersembunyi:
vij (baru) = vij (lama) + vij (j=1,2, ...p ; i =0,1,2,...,n)
9. Berhenti

Setelah selesai, jaringan dapat digunakan untuk pelatihan pengenalan pola. Dalam hal ini, hanya Propagasi Maju (langkah 4 dan 5) yang digunakan untuk menentukan keluaran jaringan. Jika fungsi aktivasi yang digunakan bukan sigmoid biner, maka langkah 4 dan 5 harus disesuaikan. Demikian juga turunan pada langkah 6 dan 7

8.14 Penelitian yang berhubungan dengan Jaringan syaraf Tiruan

Peneliti menggunakan banyak metode untuk mengekstrak dan mengklasifikasikan sinyal jantung. Dalam penelitian ini wavelet haar digunakan untuk mengekstrak karakteristik sinyal jantung. Jaringan saraf tiruan Backpropagation untuk klasifikasi sinyal jantung. Data diambil dari Physiobank yaitu MIT-BIH Arrhythmia Database dan MIT-BIH Normal Sinus Rhythm Database. Data tersebut diolah menggunakan metode Haar wavelet untuk ekstraksinya. Hasil dari metode ekstraksi ciri tersebut akan digunakan untuk proses klasifikasi. Penelitian menemukan bahwa dengan menggunakan ekstraksi ciri Wavelet Haar dan klasifikasi menggunakan Backpropagation diperoleh tingkat akurasi klasifikasi sebesar 92%.

2017 1st International Conference on Engineering and Applied Technology (ICEAT) IOP Publishing
IOP Conf. Series: Materials Science and Engineering 403 (2018) 012069 doi:10.1088/1757-899X/403/1/012069

Classification of heart signal using wavelet haar and backpropagation neural network

H Hindarto, I Anshory and A Efiyanti

Universitas Muhammadiyah Sidoarjo
Jl. Mojopahit 666B Sidoarjo
Jl. Raya Gelam 250 Candi Sidoarjo
+62-031-894 5444

hindarto@umsida.ac.id

Abstract. Researchers used many methods to extract and classify heart signals.In this study wavelet haar is use to extract characteristics of heart signals. Artificial neural networks Backpropagation for the classification of heart signals. The data is taken from Physiobank namely MIT-BIH Arrhythmia Database and MIT-BIH Normal Sinus Rhythm Database. The data is processed using Haar wavelet method for its extraction.The results of feature extraction methods will be use for the classification process. The research found that by using Wavelet Haar feature extraction and classification using Backpropagation obtained classification accuracy rate of 92%.

Keywords: Heart; electrocardiography; Wavelet Haar; backpropagation

1. Introduction

Electrocardiogram (ECG) is the electrical device to detect the activity of the human heart. The ECG is a composite of 5 waves - P, Q, R, S and T. These signals can be measured with electrode, placed on the human body. The signal from this electrode is connect to a simple electrical circuit with amplifier and analog - digital converter. Application of heart rate and heart rate detection by using an electronic circuit with a stethoscope and pulse sensor as input to detect. The pulse sensor is use to determine the heart of a person and a stethoscope is use to detect heart sound. While the series of electronics used arduino series as an interface to the monitor screen. From Test results on 10 subjects, to measure of heart rate with the pulse sensor obtained the accuracy of the tool sensory pulses by way of 99% [1].

The heart frequency can be detected by many methods and algorithms. Many heart signal detection algorithm is based on the distance between the QRS complexes. Complex QRS algorithms are from the field of artificial neural networks, genetic algorithms, wavelet transforms or bank filters [2]. In addition the next way to detect complex QRS is to use adaptive threshold [3] such as Direct method for detection of heart rate spectral signal spectral ECG [4] and the method of Short-Term Autocorrelation [5]. ECG signals can be used to diagnose heart disease, however, t ECG signals do not fully describe the heart characteristics. It is because the heart is also affected by the opening and closing of the heart valve is a factor in the conscience. In addition, there is a heart damage that is difficult to detect using an ECG, such as natural structural abnormalities or opening and closing imperfect heart valves, as well as heart murmurs or abnormal sounds [6].

There are several studies by taking data from MIT-BIH data base. Thisresearch uses Heart Rate Variability (HRV) analysis method to look for feature extraction and classification using artificial neural network classifier. Results obtained with an accuracy of 99.38% [7]. F. Yaghouby et al., In his penisilitan using Generalized Discriminant Analysis (GDA) method for extracting and using Multilayer Perceptron (MLP) method of artificial neural network grouping, the result is 100% [8]. R. Acharya et al., This study took eight classes, feature extraction used by taking from the spectral entropy, the Poincar geometry plot and the largest Lyapunov exponent (LLE). Then classification using artificial neural network method and fuzzy relationship. This research yields value. 80-85 5 [9]. L. Hussain et al., This study to look for feature extraction using HRV

Soal :

1. Lakukan pelatihan pengenalan pola fungsi logika “OR” dengan menggunakan algoritma perceptron dimana input dan output biner, dimana bobot awal = 0, bias awal = 0, learning rate = 1 dan threshold = 0.
2. Lakukan pelatihan pengenalan pola fungsi logika “OR dengan menggunakan algoritma Adaline dimana input dan output bipolar, dimana batas toleransi = 0.05, $\alpha = 0.1$.
3. Lakukan pelatihan BackPropagation untuk mengenali fungsi logika “AND” dengan input dan output biner. Dimana bobot awal = 0, bias awal = 0, learning rate = 1, dan threshold = 0.

BAB 9
Algoritma Genetika

Algoritma genetika adalah jenis algoritma optimasi, yang digunakan untuk menemukan solusi optimal untuk masalah komputasi tertentu yang memaksimalkan atau meminimalkan fungsi tertentu. Algoritma genetika mewakili satu cabang bidang studi yang disebut komputasi evolusioner, yang meniru proses biologis reproduksi dan seleksi alam untuk memecahkan solusi yang paling cocok. Seperti dalam evolusi, banyak proses algoritma genetika yang acak, namun teknik optimasi ini memungkinkan seseorang untuk mengatur tingkat pengacakan dan tingkat kontrol. Algoritma ini jauh lebih kuat dan efisien daripada pencarian acak dan algoritma pencarian lengkap, namun tidak memerlukan informasi tambahan tentang masalah yang diberikan. Fitur ini memungkinkan menemukan solusi untuk masalah yang tidak dapat ditangani oleh metode optimasi lain karena kurangnya kontinuitas, turunan, linearitas, atau fitur lainnya.

9.1 Komponen, Struktur, & Terminologi

Sejak algoritma genetika dirancang untuk mensimulasikan proses biologis, banyak terminologi yang relevan dipinjam dari biologi. Namun, entitas yang merujuk pada terminologi ini dalam algoritma genetika jauh lebih sederhana daripada rekan biologisnya. Komponen dasar yang umum untuk algoritma genetika adalah:

- Fungsi kebugaran untuk pengoptimalan
- Populasi kromosom
- Pemilihan kromosom mana yang akan bereproduksi
- Crossover untuk menghasilkan kromosom generasi berikutnya
- Mutasi acak kromosom pada generasi baru

Fungsi fitness adalah fungsi yang coba dioptimalkan oleh algoritma. Kata “fitness” diambil dari teori evolusi. Ini digunakan di sini karena fungsi kebugaran menguji dan mengukur seberapa 'cocok' setiap solusi potensial. Fungsi fitness adalah salah satu bagian yang paling penting dari algoritma, sehingga dibahas lebih rinci. Istilah kromosom mengacu pada nilai numerik atau nilai yang mewakili kandidat solusi untuk masalah yang coba dipecahkan oleh algoritma genetika. Setiap solusi kandidat dikodekan sebagai array nilai parameter, sebuah proses yang juga ditemukan dalam algoritma optimasi lainnya. Jika suatu masalah memiliki dimensi N_{par}, maka biasanya setiap kromosom dikodekan sebagai elemen array N_{par}

$$\text{kromosom} = [p1, p2, ..., pN_{par}] \quad (9.1)$$

dimana setiap p_i adalah nilai tertentu dari parameter ke-i. Terserah pencipta algoritma genetika untuk merancang bagaimana menerjemahkan ruang sampel solusi kandidat ke dalam kromosom. Salah satu pendekatan adalah dengan mengubah setiap nilai parameter menjadi string bit (urutan 1 dan 0), kemudian menggabungkan parameter end-to-end seperti gen dalam untai DNA untuk membuat kromosom. Secara historis, kromosom biasanya dikodekan dengan cara ini, dan tetap menjadi metode yang cocok untuk ruang solusi diskrit. Komputer modern memungkinkan kromosom untuk memasukkan permutasi, bilangan real, dan banyak objek lainnya; tetapi untuk saat ini kita akan fokus pada kromosom biner.

Algoritma genetika dimulai dengan pemilihan kromosom yang dipilih secara acak, yang berfungsi sebagai generasi pertama (populasi awal). Kemudian setiap kromosom dalam populasi dievaluasi dengan fungsi fitness untuk menguji seberapa baik ia menyelesaikan masalah yang dihadapi.

Operator seleksi memilih beberapa kromosom untuk reproduksi berdasarkan distribusi probabilitas yang ditentukan oleh pengguna. Semakin bugar suatu kromosom, semakin besar kemungkinan untuk dipilih. Sebagai contoh, jika f adalah fungsi fitness non-negatif, maka probabilitas bahwa kromosom C_{53} terpilih untuk bereproduksi adalah:

$$P(C_{53}) = |\frac{f(C_{53})}{\sum_{i=1}^{N_{pop}} f(C_i)}| \quad (9.2)$$

Perhatikan bahwa operator seleksi memilih kromosom dengan penggantian, sehingga kromosom yang sama dapat dipilih lebih dari satu kali. Operator crossover menyerupai persilangan biologis dan rekombinasi kromosom dalam sel meiosis. Operator ini menukar selanjutnya dari dua kromosom yang dipilih untuk membuat dua keturunan. Misalnya, jika kromosom induk

[11010111001000]and[01011101010010]

disilangkan setelah bit keempat, maka

[01010111001000]and[11011101010010]

akan menjadi keturunan mereka. Operator mutasi secara acak membalik bit individu dalam kromosom baru (mengubah 0 menjadi 1 dan sebaliknya). Biasanya mutasi terjadi dengan probabilitas yang sangat rendah, seperti 0,001. Beberapa algoritma mengimplementasikan operator mutasi sebelum operator seleksi dan crossover; ini adalah masalah preferensi. Sepintas, operator mutasi mungkin tampak tidak perlu. Faktanya, ini memainkan peran penting, bahkan jika itu sekunder dari seleksi dan crossover. Seleksi dan crossover mempertahankan informasi genetik dari kromosom yang lebih bugar, tetapi kromosom ini hanya lebih bugar relatif terhadap generasi saat ini. Hal ini dapat menyebabkan algoritma untuk berkumpul terlalu cepat dan kehilangan "bahan genetik yang berpotensi berguna (1 atau 0 di lokasi tertentu)". Dengan kata lain, algoritma dapat terjebak pada optimum lokal sebelum menemukan optimum global. Operator mutasi membantu melindungi dari masalah ini dengan mempertahankan keragaman dalam populasi, tetapi juga dapat membuat algoritma konvergen lebih lambat.

Biasanya proses seleksi, persilangan, dan mutasi berlanjut sampai jumlah keturunan sama dengan populasi awal, sehingga generasi kedua seluruhnya terdiri dari keturunan baru dan generasi pertama tergantikan seluruhnya. Kita akan melihat metode ini dalam Contoh 9.1 dan 9.2. Namun, beberapa algoritma membiarkan anggota yang sangat cocok dari generasi pertama bertahan hingga generasi kedua. Kita akan melihat metode ini dalam Contoh 9.3.

Sekarang generasi kedua diuji oleh fungsi kebugaran, dan siklus berulang. Ini adalah praktik umum untuk merekam kromosom dengan kebugaran tertinggi (bersama dengan nilai kebugarannya) dari setiap generasi, atau kromosom “terbaik sejauh ini”.

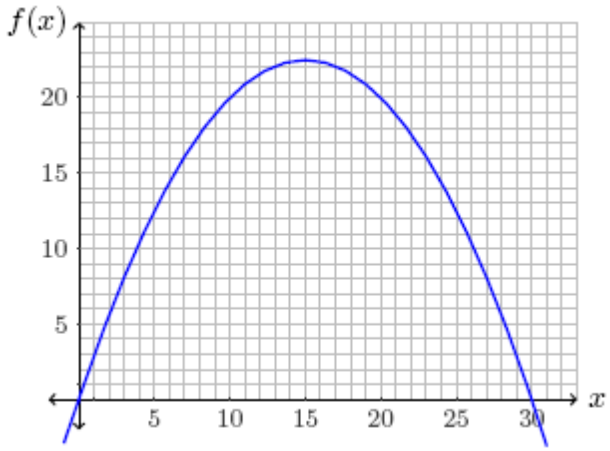
Algoritma genetika diulang sampai nilai fitness dari kromosom “terbaik sejauh ini” stabil dan tidak berubah selama beberapa generasi. Ini berarti algoritma telah konvergenuntuk solusi (s). Seluruh proses iterasi disebut run. Pada akhir setiap run biasanya ada setidaknya satu kromosom yang merupakan solusi yang sangat cocok untuk masalah aslinya. Bergantung pada bagaimana algoritma ditulis, ini bisa menjadi yang paling cocok dari semua kromosom "terbaik sejauh ini", atau yang paling cocok dari generasi terakhir.

“Kinerja” dari algoritma genetika sangat bergantung pada metode yang digunakan untuk mengkodekan solusi kandidat ke dalam kromosom dan “kriteria tertentu untuk sukses”, atau apa yang sebenarnya diukur oleh fungsi fitness. Detail penting lainnya adalah probabilitas crossover, probabilitas mutasi, ukuran populasi, dan jumlah iterasi. Nilai-nilai ini dapat disesuaikan setelah menilai kinerja algoritma pada beberapa percobaan.

Algoritma genetika digunakan dalam berbagai aplikasi. Beberapa contoh yang menonjol adalah pemrograman otomatis dan pembelajaran mesin. Cocok untuk memodelkan fenomena di bidang ekonomi, ekologi, sistem kekebalan manusia, genetika populasi, dan sistem sosial.

9.1.1 Fungsi Fitnes

Melanjutkan analogi seleksi alam dalam evolusi biologis, fungsi fitnes adalah seperti habitat dimana organisme (kandidat solusi) beradaptasi. Ini adalah satu-satunya langkah dalam algoritma yang menentukan bagaimana kromosom akan berubah dari waktu ke waktu, dan dapat berarti perbedaan antara menemukan solusi optimal dan tidak menemukan solusi sama sekali. Kinnear, editor Advances in Genetic Programming, menjelaskan bahwa “fungsi kebugaran adalah satu-satunya kesempatan yang Anda miliki untuk mengomunikasikan niat Anda pada proses hebat yang diwakili oleh pemrograman genetik. Pastikan itu mengomunikasikan dengan tepat apa yang Anda inginkan”. Sederhananya, "Anda tidak bisa terlalu berhati-hati dalam membuat" itu. Kinnear menekankan bahwa evolusi populasi akan "mengeksploitasi dengan kejam" semua "kondisi batas" dan cacat halus pada fungsi kebugaran, dan satu-satunya cara untuk mendeteksi ini adalah dengan menjalankan algoritma dan memeriksa kromosom yang dihasilkan.



Gambar 9.1 Grafik dari f(x) = $-x^2/10 + 3x$

Fungsi fitness harus lebih sensitif daripada hanya mendeteksi apa yang merupakan kromosom 'baik' versus kromosom 'buruk': perlu menilai kromosom secara akurat berdasarkan rentang nilai fitness, sehingga solusi yang agak lengkap dapat dibedakan dari solusi yang lebih lengkap. Kinnear menyebut pemberian ini sebagai “partial credit”. Penting untuk mempertimbangkan solusi parsial mana yang lebih disukai daripada solusi parsial lainnya karena itu akan menentukan arah pergerakan seluruh populasi.

9.2 Contoh Pendahuluan algoritma genetika

Bagian ini akan membahas beberapa contoh sederhana dari algoritme genetika yang sedang bekerja. Disajikan dalam urutan peningkatan kompleksitas dan dengan demikian mengurangi generalitas.

9.2.1 Memaksimalkan Fungsi Satu Variabel

Contoh ini mengadaptasi metode contoh yang disajikan dalam buku Goldberg .Pertimbangkan masalah memaksimalkan fungsi.

$$f(x) = \frac{-x^2}{10} + 3x$$

dimana x diperbolehkan untuk bervariasi antara 0 dan 31. Fungsi ini ditampilkan pada Gambar 9.1. Untuk menyelesaikan ini menggunakan algoritma genetika, kita harus mengkodekan nilai x yang mungkin sebagai kromosom. Untuk contoh ini, kami akan mengkodekan x sebagai bilangan bulat biner dengan panjang 5. Jadi kromosom untuk algoritma genetika kami akan menjadi urutan 0 dan 1 dengan panjang 5 bit, dan memiliki rentang dari 0 (00000) hingga 31 (11111).
Untuk memulai algoritma, kami memilih populasi awal 10 kromosom secara acak. Kita dapat mencapai ini dengan melempar koin yang adil 5 kali untuk setiap kromosom, membiarkan kepala menandakan 1 dan ekor menandakan 0. Populasi awal kromosom yang dihasilkan ditunjukkan pada Tabel 9.1. Selanjutnya kita ambil nilai x yang diwakili oleh setiap kromosom dan uji kecocokannya dengan fungsi kecocokan. Nilai fitness yang dihasilkan dicatat pada kolom ketiga dari Tabel 9.1.

Tabel 9.1: Populasi Awal

Nomor Kromosom	Populasi Awal	Nilai X	Nilai Fitnes f(x)	Probabilitas seleksi
1	01011	11	20.9	0.1416
2	11010	26	10.4	0.0705
3	00010	2	5.6	0.0379
4	01110	14	22.4	0.1518
5	01100	12	21.6	0.1463
6	11110	30	0	0
7	10110	22	17.6	0.1192
8	01001	9	18.9	0.1280
9	00011	3	8.1	0.0549
10	10001	17	22.1	0.1497
		Jumlah	147.6	
		Rata-rata	14.76	
		Maksimal	22.4	

Untuk memilih kromosom yang akan bereproduksi berdasarkan nilai fitnessnya, menggunakan probabilitas berikut:

$$P(kromosom\ i\ reproduksi) = |\frac{f(x_i)}{\sum_{k=1}^{10} f(x_k)}| \quad (9.3)$$

Goldberg menyamakan proses ini dengan memutar roda rolet berbobot. Karena populasi memiliki 10 kromosom dan setiap 'kawin' menghasilkan 2 keturunan, kita membutuhkan 5 perkawinan untuk menghasilkan generasi baru 10 kromosom. Kromosom yang dipilih ditampilkan pada Tabel 9.2. Untuk membuat keturunannya, titik crossover dipilih secara acak, yang ditunjukkan pada tabel sebagai garis vertikal. Perhatikan bahwa ada kemungkinan bahwa persilangan tidak terjadi, dalam hal ini keturunannya adalah salinan persis dari orang tuanya.

Tabel 9.2: Reproduksi & Generasi Kedua

Nomor Kromosom	Pasangan Kawin	Populasi Baru	Nilai x	Nilai Fitnes f(x)
5	01|100	01010	10	20
2	11|010	11100	28	5.6
4	0111|0	01111	15	22.5

8	0100|1	01000	8	17.6
9 2	0001|1 1101}0	01010 11011	10 27	20 8.1
7 4	10110 01110	10110 01110	22 14	17.6 22.4
10 8	100|01 010|01	10001 01001	17 9	22.1 18.9
			Jumlah Rata-rata Maksimal	174.8 17.48 22.5

Terakhir, setiap bit kromosom baru bermutasi dengan probabilitas rendah. Untuk contoh ini, membiarkan probabilitas mutasi menjadi 0,001. Dengan total 50 posisi bit yang ditransfer, yang diharapkan 50x0,001 = 0,05 bit untuk bermutasi. Jadi kemungkinan tidak ada bit yang bermutasi pada generasi kedua. Demi ilustrasi, telah bermutasi satu bit dalam populasi baru, yang ditunjukkan dalam huruf tebal pada Tabel 9.2.

Setelah seleksi, crossover, dan mutasi selesai, populasi baru diuji dengan fungsi fitness. Nilai fitness dari generasi kedua ini tercantum dalam Tabel 9.2. Meskipun menarik kesimpulan dari satu percobaan algoritma berdasarkan probabilitas adalah, "paling baik, bisnis yang berisiko," contoh ini menggambarkan bagaimana algoritma genetika 'berkembang' menuju solusi kandidat yang lebih bugar. Membandingkan Tabel 9.2 dengan Tabel 9.1, terlihat bahwa keduanya maksimumkebugaran dan kebugaran rata-rata populasi telah meningkat setelah hanya satu generasi.

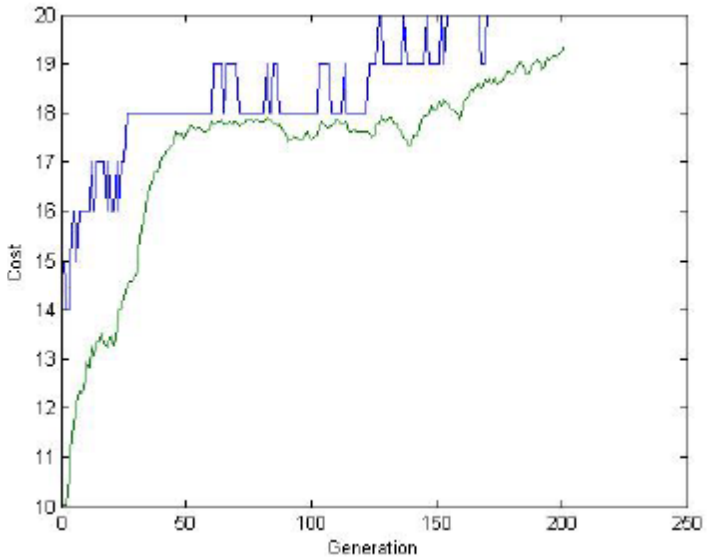
9.2.2 Contoh: Jumlah 1 dalam sebuah String

Contoh ini mengadaptasi kode Haupt untuk algoritma genetika biner ke latihan komputer pertama dari bab 1 buku teks Mitchell.Dalam contoh ini kita akan memprogram algoritma genetika lengkap menggunakan MATLAB untuk memaksimalkan jumlah 1 dalam bit string dengan panjang 20. Dengan demikian kromosom kita akan menjadi string biner dengan panjang 20, dan kromosom optimal yang kita cari adalah

[11111111111111111111] .

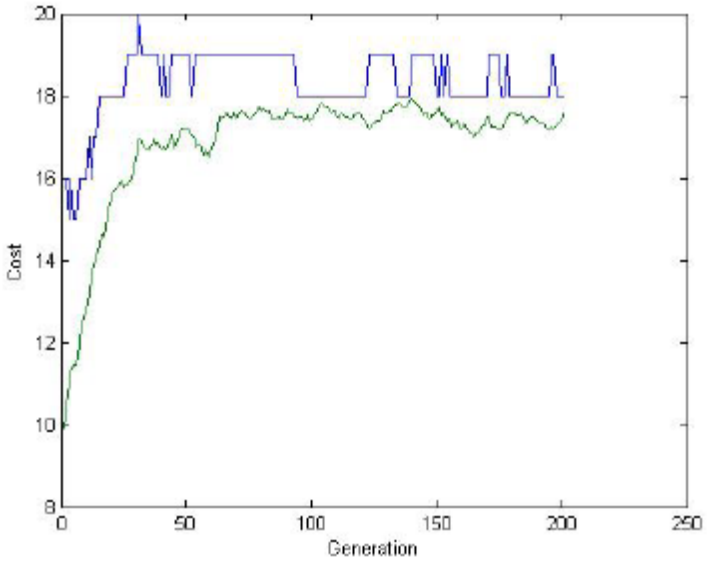
Kode MATLAB lengkap untuk algoritma ini tercantum dalam Lampiran A. Karena satu - satunya kemungkinan untuk setiap bit kromosom adalah 1 atau 0, jumlah 1 dalam kromosom setara dengan jumlah semua bit. Oleh karena itu, kami mendefinisikan fungsi fitness sebagai jumlah bit dalam setiap kromosom. Selanjutnya kita tentukan ukuran populasi menjadi 100 kromosom, tingkat mutasi menjadi 0,001, dan jumlah iterasi (generasi) menjadi 200. Demi kesederhanaan, biarkan probabilitas crossover menjadi 1 dan lokasi crossover akan selalu sama.

Gambar 9.2, 9.3, dan 9.4 menunjukkan hasil dari tiga run. Solusi terbaik yang ditemukan olehkedua run 1dan 3 adalah [11111111111111111111] . Solusi terbaik yang ditemukan oleh run 2 adalah [11101111110111111111] . Kita dapat melihat dari run 2 (3) mengapa penting untuk menjalankan seluruh algoritma lebih dari sekali karena run ini tidak menemukan solusi optimal setelah 200 generasi. Jalankan 3(4) mungkin tidak menemukan solusi optimal jika kami memutuskan untuk mengulangi algoritma kurang dari 200 kali. Hasil yang bervariasi ini disebabkan oleh sifat probabilistik dari beberapa langkah dalam algoritma genetika. Jika kita menemukan run seperti 2 dan 3 (3, 4) dalam situasi dunia nyata, akan lebih bijaksana bagi kita untuk meningkatkan jumlah iterasi untuk memastikan bahwa algoritme memang konvergen ke solusi optimal. Dalam aplikasi praktis, orang tidak tahu

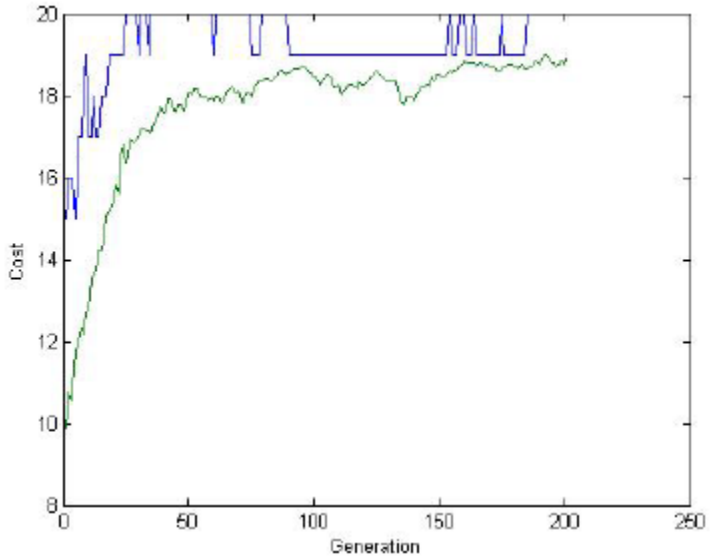


Gambar 9.2: Jalan pertama dari algoritma genetika yang memaksimalkan jumlah 1 dalam string 20 bit.

Kurva biru adalah kebugaran tertinggi, dan kurva hijau adalah kebugaran rata-rata. Solusi terbaik: [11111111111111111111] .



Gambar 9.3: Proses kedua dari algoritma genetika yang memaksimalkan jumlah 1 dalam string 20 bit. Kurva biru adalah kebugaran tertinggi, dan kurva hijau adalah kebugaran rata-rata. Solusi terbaik: [11101111110111111111] .



Gambar 9.4: Proses ketiga dari algoritma genetika yang memaksimalkan jumlah 1 dalam string 20 bit. Kurva biru adalah kebugaran tertinggi, dan kurva hijau adalah kebugaran rata-rata. Solusi terbaik: [11111111111111111111] .

seperti apa solusi optimalnya, jadi sudah menjadi praktik umum untuk melakukan beberapa proses.

9.2.3 Contoh: Algoritma Genetika Kontinu

Di sini disajikan gaya baru dari algoritma genetika, dan petunjuk penerapannya dalam topografi. Contoh ini mengadaptasi kode dan contoh untuk algoritma genetika berkelanjutan dari buku Haupt. Mungkin telah memperhatikan bahwa Contoh diatas hanya memungkinkan untuk solusi bilangan bulat, yang dapat diterima dalam kasus tersebut karena fungsi maksimum yang dimaksud adalah bilangan bulat. Tetapi bagaimana jika kita membutuhkan algoritma untuk mencari melalui rangkaian nilai? Dalam hal ini, lebih masuk akal untuk membuat setiap kromosom sebuah array bilangan real (bilangan floating-point) sebagai lawan dari array hanya 0 dan 1. Dengan cara ini, ketepatan solusi hanya dibatasi oleh komputer, bukan oleh representasi biner dari angka-angka. Faktanya, algoritma genetika kontinu ini lebih cepat daripada algoritma genetika biner karena kromosom tidak perlu didekodekan sebelum menguji kebugarannya dengan fungsi kebugaran [3]. Ingat bahwa jika masalah kita memiliki dimensi N_{par}, maka kita akan mengkodekan setiap kromosom sebagai array elemen N_{par}

Kromosom = $[p_1, p_2, ..., pN_{par}]$

di mana setiap pi adalah nilai tertentu dari parameter ke-i, hanya saja kali ini setiap pi adalah bilangan real, bukan bit.

Misalkan kita sedang mencari titik elevasi terendah pada peta topografi (hipotetis) di mana garis bujur berkisar dari 0 hingga 10, garis lintang berkisar dari 0 hingga 10, dan elevasi lanskap diberikan oleh fungsi:

$F(x,y) = x\sin(4x) + 1.1\ y \sin(2y)$

Untuk mengatasi masalah ini, akan memprogram algoritma genetika kontinu menggunakan MATLAB. Dapat dilihat bahwa f (x, y) akan menjadi fungsi fitness kita karena kita sedang mencari solusi yang meminimalkan elevasi. Karena f adalah fungsi dari dua variabel, N_{par} = 2 dan kromosom kita harus berbentuk

Kromosom = [x,y], untuk $0 \leq x \leq 10$ dan $0 \leq y \leq 10$

Mendefinisikan ukuran populasi kami menjadi 12 kromosom, tingkat mutasi menjadi 0,2, dan jumlah iterasi menjadi 100. Seperti dalam algoritma genetika biner, populasi awal dihasilkan secara acak, kecuali kali ini parameternya dapat berupa bilangan real apa pun antara 0 dan 10, bukan hanya 1 atau 0.
Dengan algoritma ini kami menunjukkan pendekatan yang sedikit berbeda untuk membangun generasi berikutnya daripada yang kami gunakan dalam contoh sebelumnya. Alih-alih mengganti seluruh populasi dengan keturunan baru, kami mempertahankan setengah lebih bugar dari populasi saat ini, dan menghasilkansetengah lainnya dari generasi baru melalui seleksi dan crossover. Tabel 9.3 dan 9.4 menunjukkan kemungkinan populasi awal dan setengahnya yang bertahan hingga generasi berikutnya. Operator seleksi hanya memilih orang tua dari sebagian kecil dari populasi yang disimpan.

Tabel 9.3: Contoh Populasi Awal

x	y	Fitnes f(x,y)
6.7874	6.9483	13.5468
7.5774	3.1710	-6.5696
7.4313	9.5022	-5.7656
3.9223	0.3445	0.3149
6.5548	4.3874	8.7209
1.7119	3.8156	5.0089
7.0605	7.6552	3.4901
0.3183	7.9520	-1.3994
2.7692	1.8687	-3.9137
0.4617	4.8976	-1.5065
0.9713	4.4559	1.7482
8.2346	6.4631	10.7287

Tabel 9.4: Populasi yang Bertahan setelah Tingkat Seleksi 50%

Rank	x	y	Fitness f(x,y)
1	7.5774	3.1710	-6.5696
2	7.4313	9.5022	-5.7656
3	2.7692	1.8687	-3.9137
4	0.4617	4.8976	-1.5065
5	0.3183	7.9520	-1.3994
6	3.9223	0.3445	0.3149

Perhatikan bahwa fungsi fitness mengambil nilai positif dan negatif, jadi kita tidak bisa langsung menggunakan jumlah nilai fitness kromosom untuk menentukan probabilitas siapa yang akan dipilih untuk bereproduksi (seperti yang kita lakukan pada Contoh 9.1). Sebagai gantinya digunakan pembobotan peringkat, berdasarkan peringkat yang ditunjukkan pada Tabel 9.4. Probabilitas bahwa kromosom di tempat ke-n akan menjadi orang tua diberikan oleh persamaan

$$P_n = \frac{N_{keep}-n+1}{\sum_{i=1}^{N_{Keep}} i} = \frac{6-n+1}{1+2+3+4+5+6} = \frac{7-n}{21}$$

jadi peluang terambilnya kromosom pertama adalah

$$P\ (C_1) = \frac{6-1+1}{1+2+3+4+5+6} = \frac{6}{21}$$

dan peluang terambilnya kromosom di urutan keenam adalah

$$P\ (C_6) = \frac{6-6+1}{1+2+3+4+5+6} = \frac{1}{21}$$

Ingatlah bahwa setiap perkawinan menghasilkan 2 keturunan, jadi kita membutuhkan 3 pasang kromosom induk untuk menghasilkan jumlah keturunan yang tepat untuk mengisi sisa generasi berikutnya.Karena kromosom kita bukan lagi bit string, kita perlu merevisi cara kerja operator crossover dan mutasi. Ada beberapa metode untuk mencapai ini; disini kita akan menggunakan metode Haupt [3]. Setelah kita memiliki dua kromosom induk m = [xm; ym] dan d = [xd; yd], pertama kita secara acak memilih salah satu parameter untuk menjadi titik crossover. Sebagai ilustrasi, misalkan x adalah titik persilangan untuk m dan d tertentu. Kemudian kami memperkenalkan nilai acak antara 0 dan 1 yang diwakili oleh ß, dan nilai-x pada pegas o adalah

$$x_{new1} = (1 - \beta)x_m + \beta x_d$$

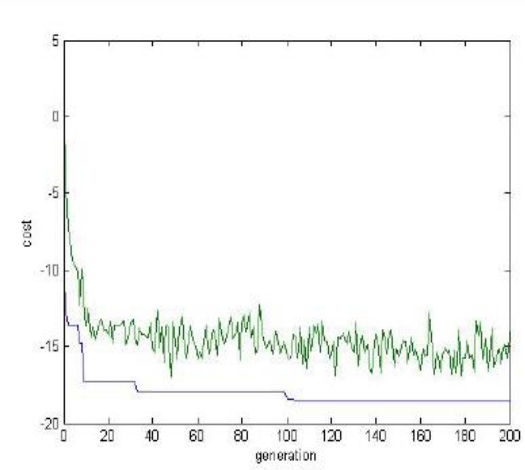
$$x_{new2} = (1 - \beta)x_d + \beta x_m$$

Parameter yang tersisa (y dalam hal ini) diwarisi langsung dari setiap orang tua, sehingga keturunan yang lengkap adalah

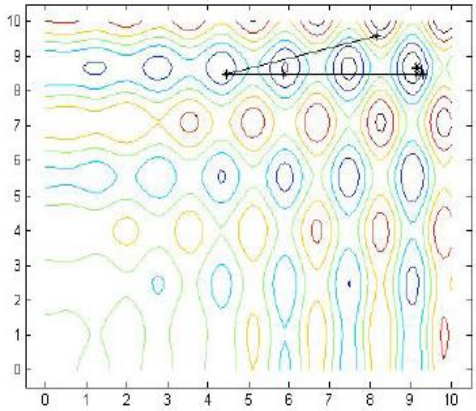
$$offsprings_1 = [x_{new1}, y_m]$$
$$offsprings_2 = [x_{new2}, y_d]$$

Jika kita menganggap bahwa kromosom1 = [7:5774 ; 3:1710] dan kromosom3 = [2:7692 ; 1:8687] dipilih sebagai pasangan orang tua. Kemudian dengan crossover di x dan -nilai ß 0,3463, keturunan mereka akan menjadi
$offsprings_1$= [(1 -0:3463) *7:5774 + 0:3463 *2:7692; 3:1710] = [5:9123; 3:1710]
$offsprings_2$= [(1 -0:3463) *2:7692 + 0:3463 *7:5774; 1:8687] = [4:4343; 1:8687] :

Seperti operator crossover, ada banyak metode berbeda untuk mengadaptasi operator mutasi ke algoritma genetika kontinu. Untuk contoh ini, ketika parameter dalam kromosom bermutasi, nilainya diganti dengan angka acak antara 0 dan 10.



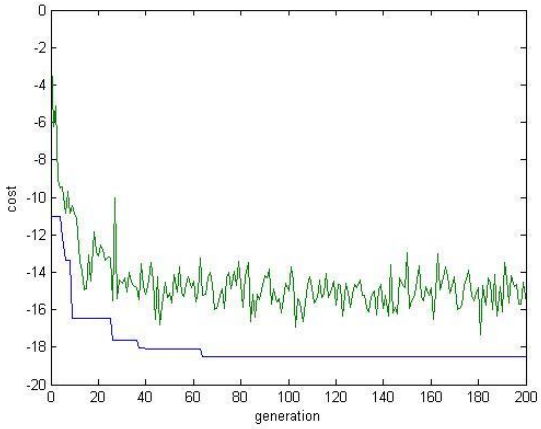
a) Kurva biru adalah elevasi terendah, kurva hijau adalah elevasi rata-rata.



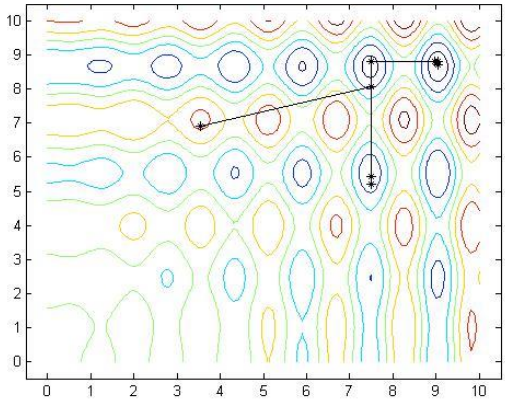
b) Peta kontur yang menunjukkan `jalur' kromosom yang paling cocok di setiap generasi ke generasi berikutnya.

Gambar 9.5: Jalan pertama dari algoritma genetika kontinu menemukan titik elevasi terendah dari fungsi f(x; y) = x sin(4x) + 1:1y sin(2y) di daerah 0 ≤x ≤10 dan 0 y 10: Dengan elevasi -18,5519, solusi terbaik adalah [9:0449; 8:6643]:

Gambar 9.5, 9.6, dan 9.7 menunjukkan hasil dari tiga run. Kita dapat melihat di ketiga run bahwa algoritma menemukan minimum jauh sebelum mencapai generasi ke-200. Dalam kasus khusus ini, jumlah generasi yang berlebihan tidak menjadi masalah karena algoritmenya sederhana dan efisiensinya tidak terlalu terpengaruh. Perhatikan bahwa karena anggota terbaik dari generasi tertentu bertahan hingga generasi berikutnya, nilai f terbaik selalu meningkat atau tetap sama. Kami juga melihat bahwa solusi akhir setiap run sedikit berbeda dari yang lain. Ini tidak mengherankan karena parameter kami adalah variabel kontinu, dan dengan demikian memiliki jumlah kemungkinan nilai tak terbatas yang mendekati nilai minimum aktual.

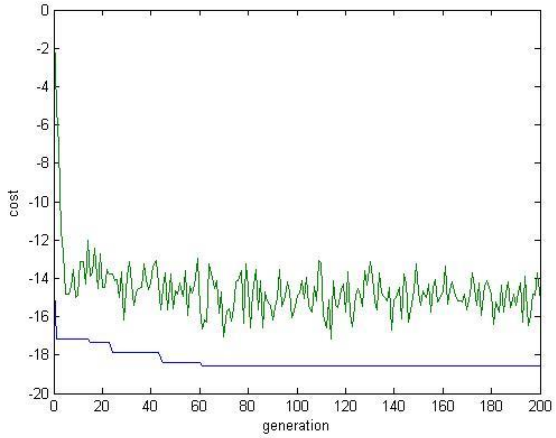


(a) Kurva biru adalah elevasi terendah, kurva hijau adalah av

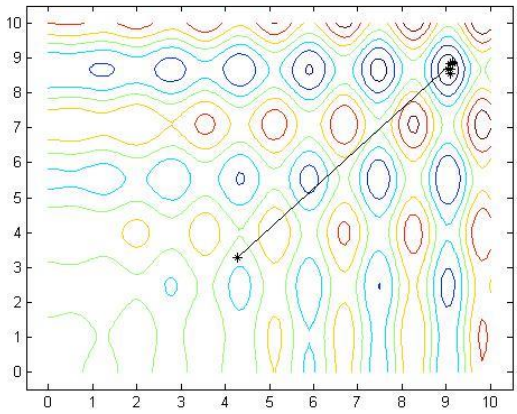


(b) Peta kontur menunjukkan 'jalur' kromosom yang paling cocok di setiap generasi ke elevasi berikutnya

Gambar 9.6: Proses kedua dari algoritma genetika kontinu menemukan titik elevasi terendah dari fungsi f (x, y) = x sin(4x) + 1.1y sin(2y) pada daerah 0≤ x≤ 10 dan 0 ≤y ≤10. Dengan elevasi -18,5227, solusi terbaik adalah [9.0386, 8.709].



(a) Kurva biru adalah elevasi terendah, kurva hijau adalah elevasi rata-rata.



(b) Peta kontur menunjukkan 'jalur' kromosom yang paling cocok di setiap generasi ke generasi berikutnya.

Gambar 7: Proses ketiga dari algoritma genetika kontinu menemukan titik elevasi terendah dari fungsi f (x, y) = x sin(4x) + 1.1y sin(2y) pada daerah 0 ≤x ≤10 dan 0≤ y≤ 10. Dengan elevasi -18,5455, solusi terbaik adalah [9.0327, 8.6865].

9.3 Konteks Sejarah

Sejak tahun 1950-an, para ilmuwan telah mempelajari kecerdasan buatan dan komputasi evolusioner, mencoba menulis program komputer yang dapat mensimulasikan proses yang terjadi di alam. Pada tahun 1953, Nils Barricelli diundang ke Princeton untuk mempelajari kecerdasan buatan. Dia menggunakan komputer digital yang baru ditemukan untuk menulis perangkat lunak yang meniru reproduksi dan mutasi alami. Tujuan Barricelli bukanlah untuk memecahkan masalah optimasi atau mensimulasikan evolusi biologis, melainkan untuk menciptakan kehidupan buatan. Dia menciptakan perangkat lunak algoritma genetika pertama, dan karyanya diterbitkan dalam bahasa Italia pada tahun 1954. Karya Barricelli diikuti pada tahun 1957 oleh Alexander Fraser, seorang ahli biologi dari London. Dia memiliki ide untuk membuat model evolusi komputer, karena mengamatinya secara langsung akan membutuhkan jutaan tahun. Dia adalah orang pertama yang menggunakan pemrograman komputer semata-mata untuk mempelajari evolusi. Banyak ahli biologi mengikuti jejaknya pada akhir 1950-an dan 1960-an.

Dalam bukunya, Mitchell menyatakan bahwa John Holland menemukan algoritma genetika pada tahun 1960-an, atau setidaknya versi tertentu yang sekarang dikenal dengan judul khusus itu. Versi Holland melibatkan simulasi 'survival of the fittest' Darwinian, serta proses crossover, rekombinasi, mutasi, dan inversi yang terjadi dalam genetika. Metode berbasis populasi ini merupakan inovasi besar. Algoritma genetika sebelumnya hanya menggunakan mutasi sebagai penggerak evolusi. Holland adalah seorang profesor psikologi, ilmu komputer, dan teknik listrik di Universitas Michigan. Dia didorong oleh pengejaran untuk memahami bagaimana "sistem beradaptasi dengan lingkungannya". Setelah bertahun-tahun berkolaborasi dengan mahasiswa dan rekan, ia menerbitkan bukunya yang terkenal Adaptasi dalam

Sistem Alami dan Buatan pada tahun 1975. Dalam buku ini, Holland menyajikan algoritma genetika sebagai "abstraksi evolusi biologis dan memberikan kerangka teoritis untuk adaptasi" di bawah algoritma genetika. Buku ini adalah yang pertama mengusulkan landasan teoretis untuk evolusi komputasi, dan tetap menjadi dasar dari semua karya teoretis tentang algoritma genetika hingga saat ini. Ini menjadi klasik karena demonstrasi matematika di balik evolusi. Pada tahun yang sama salah satu mahasiswa doktoral Belanda, Kenneth De Jong, mempresentasikan yang pertama studi komprehensif tentang penggunaan algoritma genetika untuk memecahkan masalah optimasi sebagai disertasi doktornya. Karyanya sangat teliti sehingga selama bertahun-tahun, makalah apa pun tentang algoritma genetika yang tidak menyertakan contoh patokannya dianggap "tidak memadai".

Penelitian tentang algoritma genetika meningkat pesat pada 1970-an dan 1980-an. Hal ini antara lain disebabkan oleh kemajuan teknologi. Ilmuwan komputer juga mulai menyadari keterbatasan pemrograman konvensional dan metode optimasi tradisional untuk memecahkan masalah yang kompleks. Para peneliti menemukan bahwa algoritme genetika adalah cara untuk menemukan solusi atas masalah yang tidak dapat dipecahkan oleh metode lain. Algoritme genetika dapat secara bersamaan menguji banyak titik dari seluruh ruang solusi, mengoptimalkan dengan parameter diskrit atau kontinu, menyediakan beberapa parameter optimal alih-alih solusi tunggal, dan bekerja dengan banyak jenis data yang berbeda. Keunggulan ini memungkinkan algoritme genetika untuk "menghasilkan hasil yang menakjubkan"ketika metode optimasi tradisional gagal total".

Ketika kami mengatakan metode optimasi "tradisional", kami mengacu pada tiga jenis utama: berbasis kalkulus, pencarian lengkap, dan acak. Metode optimasi berbasis kalkulus datang dalam dua kategori: langsung dan tidak langsung. Metode langsung 'melompat ke' fungsi tujuan dan mengikuti arah gradien menuju nilai maksimum atau minimum lokal. Ini juga dikenal sebagai metode "pendakian bukit" atau "pendakian gradien". Metode tidak langsung mengambil gradien dari fungsi tujuan, menetapkannya sama dengan nol, kemudian menyelesaikan himpunan persamaan yang dihasilkan. Meskipun teknik berbasis kalkulus ini telah dipelajari secara ekstensif dan ditingkatkan, mereka masih memiliki masalah yang tidak dapat diatasi. Pertama, mereka hanya mencari local optima, yang membuat mereka tidak berguna jika kita tidak mengetahui lingkungan dari optimum global atau jika ada local optima lain di dekatnya (dan kita biasanya tidak tahu). Kedua, metode ini membutuhkan keberadaan turunan, dan ini hampir tidak pernah terjadi dalam aplikasi praktis.

Algoritme pencarian lengkap melakukan persis seperti itu—pencarian lengkap. Algoritma ini membutuhkan "ruang pencarian terbatas, atau ruang pencarian tak terbatas diskrit" dari nilai-nilai yang mungkin untuk fungsi tujuan. Kemudian mereka menguji setiap nilai, satu per satu, untuk menemukan maksimum atau minimum. Meskipun metode ini sederhana dan dengan demikian "menarik", ini adalah yang paling tidak efisien dari semua algoritma optimasi. Dalam masalah praktis, ruang pencarian terlalu luas untuk menguji setiap kemungkinan "satu per satu dan masih memiliki kesempatan untuk menggunakan informasi [yang dihasilkan] untuk beberapa tujuan praktis".

Algoritma pencarian acak menjadi semakin populer karena orang menyadari kekurangan algoritma pencarian berbasis kalkulus dan lengkap. Gaya algoritma ini secara acak memilih beberapa pengambilan sampel yang representatif dari ruang pencarian, dan menemukan nilai optimal dalam pengambilan sampel tersebut. Meskipun lebih cepat dari pencarian yang lengkap, metode ini "dapat diharapkan untuk melakukan tidak lebih baik" daripada pencarian yang lengkap. Menggunakan jenis algoritme ini berarti bahwa kita membiarkannya secara kebetulan apakah kita akan mendekati solusi optimal atau bermil-mil jauhnya darinya.

Algoritma genetika memiliki banyak keunggulan dibandingkan metode tradisional ini. Tidak seperti metode berbasis kalkulus seperti hill-climbing, algoritma genetika berkembang dari populasi solusi kandidat, bukan dari satu nilai. Ini sangat mengurangi kemungkinan menemukan optimum lokal daripada

optimum global. Algoritme genetika tidak memerlukan informasi tambahan (seperti turunan) yang tidak terkait dengan nilai solusi yang mungkin itu sendiri. Satu-satunya mekanisme yang memandu pencarian mereka adalah nilai fitness numerik dari solusi kandidat, berdasarkan definisi fitness dari pembuatnya. Ini memungkinkan mereka berfungsi ketika ruang pencarian berisik, nonlinier, dan turunannya bahkan tidak ada. Ini juga membuatnya dapat diterapkan di lebih banyak situasi daripada algoritme tradisional, dan mereka dapat disesuaikan di setiap situasi berdasarkan apakah akurasi atau efisiensi lebih penting.

9.4 Masalah Praktis

Pada bagian ini kami menyajikan masing-masing dari dua masalah optimasi klasik dengan algoritma genetika. Kami akan memprogram algoritma ini menggunakan MATLAB.

9.4.1 The Traveling Salesman Problem

Ini mungkin masalah optimasi kombinatorial yang paling terkenal, praktis, dan dipelajari secara luas (dengan kombinatorial yang kami maksud adalah melibatkan permutasi—mengatur sejumlah item dalam urutan yang berbeda). Ini memiliki aplikasi dalam "robotika, pengeboran papan sirkuit, pengelasan, manufaktur, transportasi, dan banyak bidang lainnya". Misalnya, "papan sirkuit dapat memiliki puluhan ribu lubang, dan bor perlu diprogram untuk mengunjungi masing-masing lubang tersebut sambil meminimalkan beberapa fungsi biaya (waktu atau energi, misalnya)".

Dalam masalah ini, kami berasumsi bahwa ada seorang salesman keliling yang perlu mengunjungi n kota melalui rute sesingkat mungkin. Dia mengunjungi setiap kota tepat satu kali, lalu kembali ke kota tempat dia memulai. Oleh karena itu solusi kandidat akan menjadi daftar semua kota dalam urutan yang dia kunjungi. Kami menyatakan kota-kota sebagai kota 1, kota 2, kota 3, . . . , kota n; di mana kota 1 adalah titik awalnya. Dengan demikian kromosom untuk algoritma genetika akan menjadi permutasi yang berbeda dari bilangan bulat 1 sampai n.

Ada banyak variasi dari masalah travelling salesman, tetapi untuk tujuan kita, kita membuat asumsi berikut. Kita asumsikan jarak d dari kota ci ke kota cj adalah d(ci, cj) = d(cj, ci) untuk semua i [1, n] dan j [1, n]. Kita dapat melihat bagaimana dalam beberapa kasus dunia nyata ini tidak benar: jalan raya dalam satu arah mungkin sedikit lebih panjang dari jalan raya bepergian ke arah yang berlawanan, atau mungkin lebih mahal untuk berkendara menanjak daripada menuruni bukit (umumnya biaya perjalanan termasuk dalam apa yang kita sebut "jarak"). Kasus-kasus tersebut disebut masalah travelling salesman “asimetris”. Karena penjual berakhir di kota tempat dia memulai, ini adalah masalah penjual keliling yang "tertutup". Pada variasi “terbuka”, penjual mengunjungi semua kota tepat satu kali, sehingga dia tidak mengakhiri perjalanan di mana dia memulai.

9.4.2 Memodifikasi Crossover dan Mutasi untuk Masalah Permutasi

Berdasarkan bagaimana kami telah mengkodekan kandidat solusi ke dalam kromosom, operator crossover dan mutasi seperti yang kami definisikan untuk algoritma genetika biner di awal makalah ini tidak akan berfungsi karena setiap kota perlu direpresentasikan sekali dan hanya sekali. Untuk melihat mengapa hal ini terjadi, pertimbangkan dua kromosom induk dengan panjang 5 dan keturunannya setelah crossover normal:

Parents	Normal Crossover	Offspring (faulty)
3 5 1 2 4	3 5 | 1 2 4	3 5 5 3 2
1 4 5 3 2	1 4 | 5 3 2	1 4 1 2 4

Keturunan ini bukan permutasi yang valid dari bilangan bulat dari 1 hingga 5 karena mereka kehilangan beberapa angka dan mengulangi yang lain. Untuk mengatasi kendala ini, kami akan menggunakan teknik yang disebut crossover "siklus" dalam algoritma genetika kami. Ada beberapa cara lain untuk memodifikasi crossover untuk mengakomodasi permutasi, yang dibahas di Goldberg dan Haupt. Crossover siklus diilustrasikan dengan kromosom dengan panjang 6 pada Tabel 5.

Tabel 9.5. Contoh siklus Crossover

Orang Tua	Offsprings (step 1)	Offsprings (step 2)	Offsprings (step 3)	Offsprings (step 4)
415|3|26 346|2|15	415|2|26 3463|1|5	4|1|5216 3|4|6325	|4|45216 |3|16325	345216 416325

Pertama kita memilih lokasi acak dalam panjang kromosom. Kedua kromosom induk bertukar bilangan bulat di lokasi ini (ditandai dengan batang pada Tabel 5) untuk membuat keturunannya. Kecuali jika bilangan bulat yang ditukar memiliki nilai yang sama, setiap keturunan memiliki bilangan bulat duplikat. Kemudian kami mengganti bilangan bulat duplikat pada keturunan pertama dengan apa pun yang berada di lokasi yang sama pada keturunan kedua. Artinya sekarang ada duplikat dari bilangan bulat yang berbeda, sehingga proses berulang sampai tidak ada duplikat pada keturunan pertama (atau keturunan kedua). Sekarang setiap keturunan memiliki tepat satu dari setiap bilangan bulat, sehingga keduanya adalah permutasi yang valid.
Operator mutasi yang dimodifikasi secara acak memilih dua bilangan bulat dalam kromosom dari generasi baru dan menukarnya. Operator ini biasanya masih terjadi dengan probabilitas rendah.

9.4.3 Menggunakan Algoritma Genetika

Pada bagian ini kami membuat algoritma genetika untuk menyelesaikan masalah travelling salesman. Untuk melakukannya, kami mengadaptasi dan memperbaiki kode Haupt untuk algoritme genetika permutasi dan untuk fungsi kebugaran dari masalah salesman keliling. Kode MATLAB lengkap untuk algoritma ini tercantum dalam Lampiran C.
Biarkan ada 20 kota. Untuk merepresentasikannya, kami secara acak menempatkan 20 titik pada bidang xy dalam kotak 1 × 1 antara (0, 0) dan (1, 1). Sebelumnya, kami mendefinisikan kromosom kami sebagai permutasi dari bilangan bulat 1 hingga 20.
Misalkan n kota diurutkan dalam urutan c1 → c2 → • • • → cn dalam suatu
besar sekali. Karena kita mencoba meminimalkan total jarak yang ditempuh salesman, fungsi fitnessnya adalah total jarak D:

$$D = \sum_{k=1}^{n} d(c_k, c_{k+1})$$

dimana kota n+1 = kota 1 (kota awal) [3]. Jika kita biarkan (xi, yi) menyatakan koordinat kota ci, maka fungsi fitness menjadi

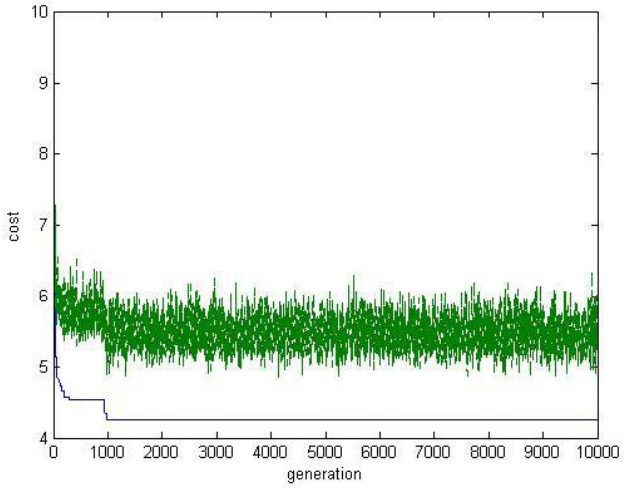
$$D = \sum_{k=1}^{n} \sqrt{(x_{k+1} - x_k)^2 + (y_{k+1} - y_k)^2} \ .$$

Selanjutnya kita mendefinisikan ukuran populasi menjadi 20. Dengan algoritma ini kita menggunakan pendekatan yang sama untuk membangun generasi berikutnya yang kita gunakan pada Contoh 2.3. Alih-alih mengganti seluruh populasi dengan keturunan baru, kami mempertahankan separuh yang lebih bugar dari populasi saat ini, dan menghasilkan separuh lainnya dari generasi baru melalui seleksi dan persilangan. Ingat bahwa operator seleksi hanya memilih dari sebagian kecil dari populasi yang disimpan. Kami memodifikasi operator crossover dan mutasi di atas (4.1.1). Pada algoritma ini, operator mutasi bekerja pada setiap anggota generasi baru dengan probabilitas 0,05.
Titik di mana algoritma ini paling menyimpang dari versi asli Haupt adalah dihitung distribusi probabilitas untuk operator seleksi, ditunjukkan pada Daftar 4.1.

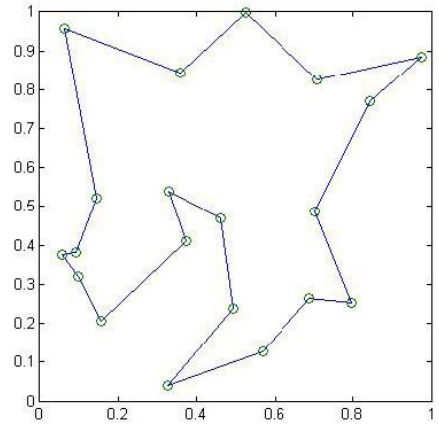
```
for ii =2: keep
odds =[ odds ii* ones (1, ii )];
end
Nodds = length ( odds );
odds =keep - odds +1;
```

Daftar 4.1: Menentukan probabilitas untuk operator seleksi berdasarkan peringkat kecocokan dari 10 kromosom teratas.

Apa yang telah kami lakukan di sini untuk memodifikasi versi Haupt adalah menambahkan baris terakhir, yang secara efektif membalikkan distribusi probabilitas kumpulan kromosom yang dipilih untuk reproduksi. Sebelum titik ini dalam kode, generasi kromosom saat ini diurutkan dari kebugaran tertinggi ke kebugaran terendah (dari 1 hingga 20), dan hanya setengah yang lebih bugar (10 kromosom) yang disimpan. Odds variabel dalam Daftar 4.1 adalah vektor yang berisi bilangan bulat dari 1 sampai 10, dengan sepuluh contoh bilangan bulat 1, sembilan contoh bilangan bulat 2, hingga satu contoh bilangan bulat 10. Sebuah kromosom kemudian dipilih untuk menjadi induk dengan memilih bilangan bulat secara acak dari peluang vektor. Oleh karena itu kita melihat bahwa peluang terpilihnya kromosom yang paling cocok adalah 10 = 2 , dan peluang terpilihnya kromosom yang paling tidak cocok adalah 1 .
55 11 55
Tanpa baris kode baru yang kami tambahkan, peluang akan memiliki sepuluh 10, sembilan 9, dll. Ini akan memberikan kromosom yang paling tidak cocok dengan probabilitas tertinggi untuk menjadi orang tua, dan sebaliknya.
Terakhir, kami menentukan jumlah iterasi menjadi 10.000.
Gambar 8, 9, 10, 11, dan 12 menunjukkan lima dari beberapa run. Karena kita tidak mengetahui secara apriori apa solusi optimalnya, penting untuk menjalankan seluruh algoritma beberapa kali hingga hasil terbaik muncul kembali beberapa kali. Kita dapat melihat dari Gambar 9 dan Gambar 11 bahwa kita bisa saja tertipu dengan berpikir bahwa 4.1816 adalah jarak terpendek yang mungkin jika kita tidak menjalankan algoritma cukup lama agar hasil 4.1211 muncul. Untungnya, 4.1211 muncul setelah hanya tiga kali run (Gambar 10). Beberapa putaran lagi mengkonfirmasi bahwa tidak ada

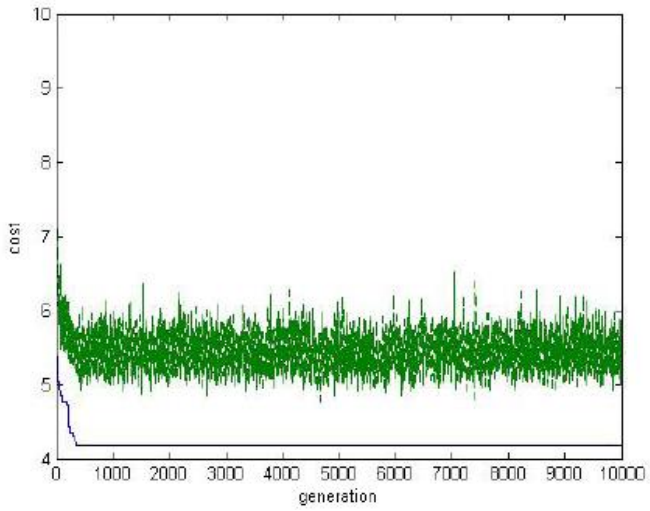


a.. Kurva biru adalah jarak terpendek, kurva hijau adalah jarak rata-rata.

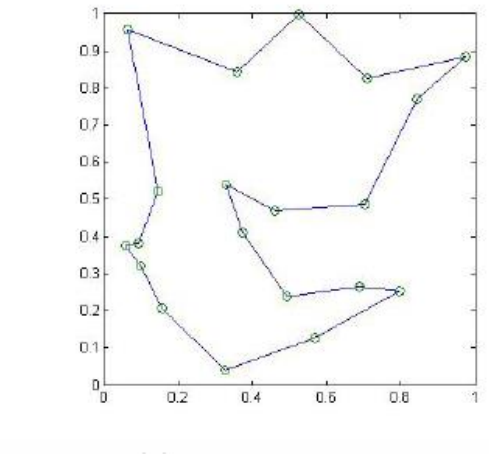


b) Grafik rute.

Gambar 9.8: Proses pertama dari algoritma genetika meminimalkan jarak untuk melakukan perjalanan dalam lingkaran tertutup ke 20 kota. Dengan jarak 4.2547, solusi terbaik adalah [13, 15, 4, 18, 11, 17, 10, 14, 1, 3, 19, 12, 5, 20, 8, 2, 9, 7, 16, 6].

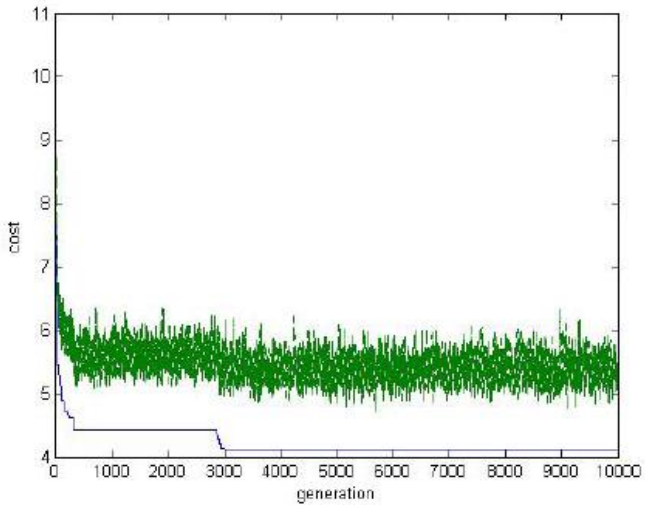


a.. Kurva biru adalah jarak terpendek, kurva hijau adalah jarak rata-rata.

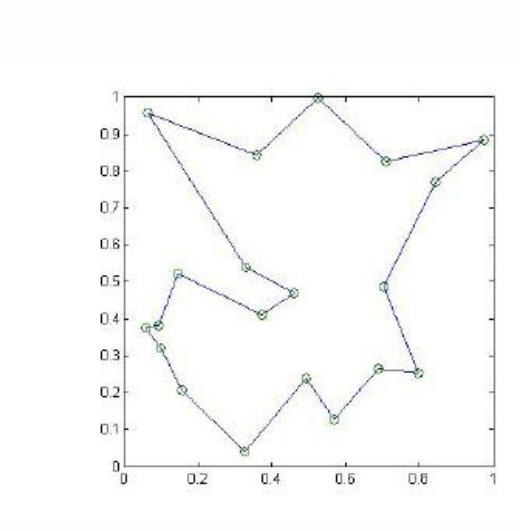


b) Grafik rute.

Gambar 9.9: Proses kedua dari algoritma genetika meminimalkan jarak untuk melakukan perjalanan dalam lingkaran tertutup ke 20 kota. Dengan jarak 4.1816, solusi terbaik adalah [4, 18, 11, 17, 10, 14, 1, 3, 19, 2, 9, 16, 7, 8, 12, 5, 20, 6, 13, 15].

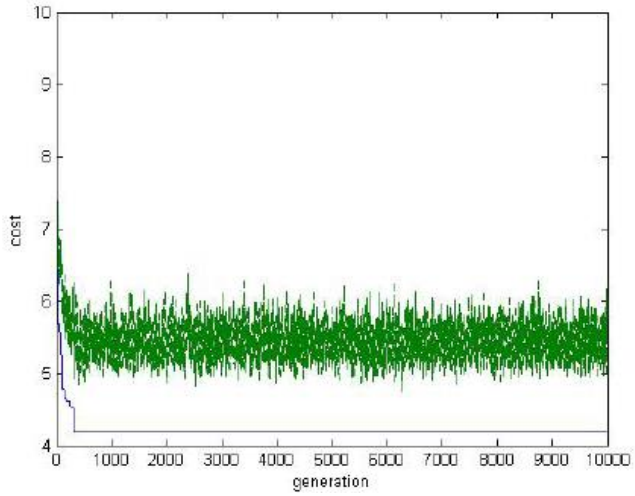


a.. Kurva biru adalah jarak terpendek, kurva hijau adalah jarak rata-rata.

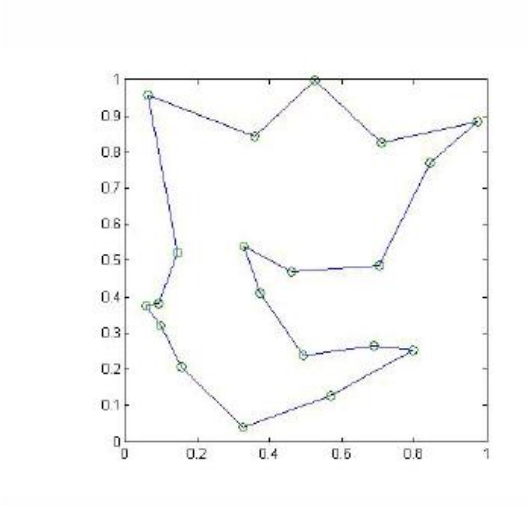


b) Grafik rute.

Gambar 9.10: Proses ketiga dari algoritma genetika meminimalkan jarak untuk melakukan perjalanan dalam lingkaran tertutup ke 20 kota. Dengan jarak 4.1211, solusi terbaik adalah [7, 9, 8, 2, 19, 3, 1, 14, 10, 12, 20, 5, 17, 11, 18, 4, 15, 13, 6, 16].

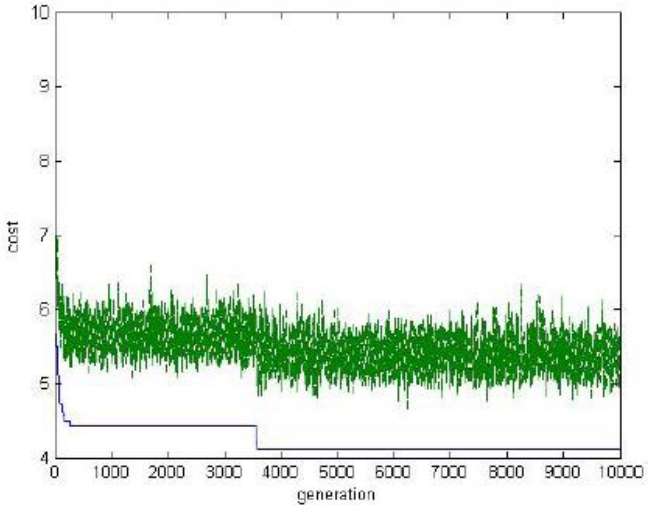


a.. Kurva biru adalah jarak terpendek, kurva hijau adalah jarak rata-rata.

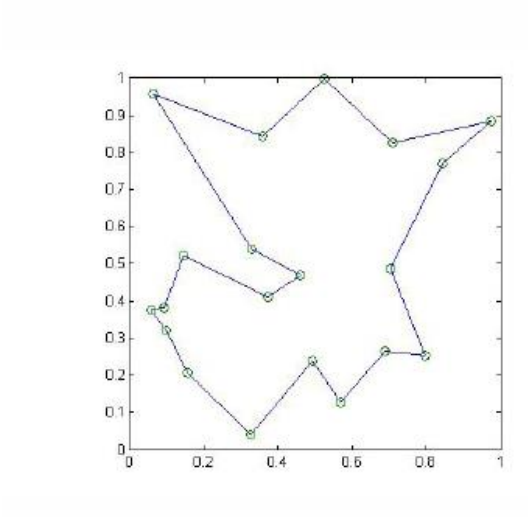


b) Grafik rute.

Gambar 9.11: Proses keempat dari algoritma genetika meminimalkan jarak untuk melakukan perjalanan dalam lingkaran tertutup ke 20 kota. Dengan jarak 4.1816, solusi terbaik adalah [10, 14, 1, 3, 19, 2, 9, 16, 7, 8, 12, 5, 20, 6, 13, 15, 4, 18, 11, 17].



a.. Kurva biru adalah jarak terpendek, kurva hijau adalah jarak rata-rata.



b) Grafik rute.

Gambar 9.12: Urutan kesebelas dari algoritma genetika meminimalkan jarak untuk melakukan perjalanan dalam lingkaran tertutup ke 20 kota. Dengan jarak 4.1211, solusi terbaik adalah [20, 12, 10, 14, 1, 3, 19, 2, 8, 9, 7, 16, 6, 13, 15, 4, 18, 11, 17, 5].

jarak lebih pendek dari 4.1211, jadi solusi optimalnya adalah

[7, 9, 8, 2, 19, 3, 1, 14, 10, 12, 20, 5, 17, 11, 18, 4, 15, 13, 6, 16] .

Run 11 (Gambar 9.12) adalah salah satu run yang mengkonfirmasi hal ini. Perhatikan bahwa solusi optimal yang ditemukan pada run 11 ekuivalen dengan run 3 meskipun kromosomnya merupakan permutasi yang berbeda. Kromosom ini dimulai di kota yang berbeda, tetapi ini tidak mempengaruhi jarak karena jalur penjual adalah loop tertutup. Kromosom ini juga mencantumkan kota-kota dalam urutan yang berlawanan dari urutan 3, tetapi ingatlah bahwa

d(ci, cj) = d(cj, ci) untuk kota ci, cj

jadi ini juga tidak mempengaruhi jarak.

9.4.4 Menggunakan Algoritma Tradisional

Mengapa masalah salesman keliling begitu sulit sebelum algoritma genetika ditemukan? Mari kita pertimbangkan untuk menggunakan algoritma pencarian lengkap untuk menyelesaikannya. Perhatikan bahwa karena penjual melakukan perjalanan dalam lingkaran tertutup dan arah perjalanan antar kota tidak menjadi masalah (permutasi melingkar bebas), jumlah total solusi yang mungkin adalah

$$S = \frac{(n-1)!}{2} = \frac{19!}{2} = 60,822,550,204,416,000$$

Jumlah ini terlalu besar untuk menguji setiap kandidat bahkan dengan nilai n yang lebih kecil dari 20. Jumlah waktu yang diperlukan untuk algoritma pencarian yang lengkap untuk menemukan solusi terbaik sepenuhnya meniadakan kegunaan jawaban apa pun yang ditemukannya.

9.4.5 Masalah Ransel

Di sini kami menyajikan masalah klasik lainnya, yang digunakan dalam kriptografi, menentukan cara terbaik untuk memanen bahan mentah, dan banyak aplikasi lainnya.
Kali ini kami hanya menyediakan metode yang mungkin untuk menerjemahkan masalah ke dalam fungsi kebugaran dan kromosom, dan cara baru untuk mendistribusikan probabilitas untuk operator seleksi. Kami mengundang pembaca untuk mengisi sisanya dan membuat algoritma lengkap untuk menyelesaikannya. Contoh ini diadaptasi dari buku kerja Steeb.

Misalkan Anda seorang pejalan kaki yang merencanakan perjalanan backpacking, dan Anda dapat membawa beban tidak lebih dari 20kg dengan nyaman. Setelah Anda memilih semua perbekalan yang ingin Anda bawa, Anda menemukan bahwa semuanya melebihi 20kg. Kemudian Anda menetapkan nilai untuk masing-masing dari 12 item, yang ditunjukkan pada Tabel 6, untuk membantu Anda memilih mana yang akan diambil. Barang mana yang harus Anda bawa untuk memaksimalkan nilai barang yang Anda bawa, tanpa melebihi 20kg?

Salah satu cara untuk menerjemahkan item ke dalam kromosom adalah dengan membuat string bit dengan panjang 12, di mana setiap posisi bit mewakili item tertentu. Biarkan 1 menunjukkan bahwa Anda menyertakan item tertentu, dan 0 menunjukkan bahwa Anda mengecualikannya.

Tabel 9.6: Nilai & Berat Perlengkapan Berkemah

Item	Nilai	Berat
pengusir serangga	12	2
kompor kamp	5	4
kantin (penuh)	10	7
pakaian	11	5
makanan kering	50	3
kit pertolongan pertama	15	3
lampu sorot	6	2
novel	4	2
perlengkapan hujan	5	2
kantong tidur	25	3
tenda	20	11
saringan air	30	1

Berdasarkan ini, fungsi kebugaran Anda harus mengetahui bobot dan nilai tetap yang terkait dengan setiap posisi bit, karena kromosom tidak membawa informasi ini. Fungsi kebugaran Anda juga harus dapat menjumlahkan dua jumlah secara bersamaan: jumlah nilai item untuk menentukan kesesuaian, dan jumlah bobot item untuk memenuhi batas bobot. Sementara fungsi fitness menjumlahkan nilai dan bobot, salah satu cara untuk memastikan bahwa kromosom tetap berada di dalam batas bobot adalah dengan segera menghentikan penjumlahan dan menetapkan nilai fitness 0 atau -1 jika berat kromosom melebihi 20kg.

Sejauh ini kita telah membahas dua metode untuk membuat distribusi probabilitas dari operator seleksi: menggunakan pecahan dari total fitness (dalam Bagian 1 dan Contoh 2.1) dan pembobotan peringkat (dalam Contoh 2.3 dan Bagian 4.1.2). Strategi lain menggunakan fungsi softmax untuk membuat distribusi yang mirip dengan metode pertama kami, kecuali jika beberapa (atau semua) nilai fitness negatif. Alih-alih membagi nilai kebugaran kromosom dengan jumlah nilai kebugaran seluruh populasi,

eksponensial nilai kebugaran kromosom dibagi dengan jumlah eksponensial nilai kebugaran populasi. Untuk mengilustrasikan, probabilitas bahwa kromosom C53 dipilih untuk bereproduksi adalah:

$$P\ (C_{53}) = \left| \frac{e^{f\ (C_{53})}}{\sum_{i=1}^{N_{pop}} e^{f(C_i)}} \right|$$

Soal :

1. Jelaskan Urutan proses dari Algoritma Genetika
2. Jelaskan Operasi – operasi pada algoritma genetika :
   a. Kromosom
   b. Mutasi
   c. Crossover
   d. Seleksi
   e. Roulette whell
3. Diketahui data binary algoritma genetika sebagai berikut :

   $$Kromosom = \left| \frac{1111001001001101111 1}{gene\ 1 \quad / \quad gen\ 2} \right|$$

   Variabel 1 minimum = 0
   Variabel 1 maksimum = 10
   Variabel 2 minimum = - 6
   Variabel 1 maksimum = 6
   Lakukan decoding untuk nilai kromosom tersebut

DAFTAR PUSTAKA

[1] Eleanor Bird, J. Fox-Skelly, N. Jenner, R. Larbey, E. Weitkamp, and A. Winfield, *The ethics of artificial intelligence Issues and initiatives*, no. March. 2020.

[2] P. E. Ekmekci and B. Arda, "History of Artificial Intelligence," *SpringerBriefs in Ethics*, no. December, pp. 1–15, 2020, doi: 10.1007/978-3-030-52448-7_1.

[3] S. Legg and M. Hutter, "Universal intelligence: A definition of machine intelligence," *Minds Mach.*, vol. 17, no. 4, pp. 391–444, 2007, doi: 10.1007/s11023-007-9079-x.

[4] A. Grignolio, V. Fortunati, G. Levi, and R. Dulbecco, *Grignolio A ., Fortunati V ._ Memory as the Origin of Biological and Cultural Identity in the Work of Samuel Butler _ 2009*, no. May 2014. 2009.

[5] C. A. Ward, "Samuel Butler," *Notes Queries*, vol. s6-X, no. 251, pp. 301–302, 1884, doi: 10.1093/nq/s6-X.251.301.

[6] J. Bing, "The riddle Prague , Č apek and the Golem," vol. 3, no. 3, pp. 197–207, 2008.

[7] J. Mańdziuk, "Some thoughts on using computational intelligence methods in classical mind board games," *Proc. Int. Jt. Conf. Neural Networks*, no. December, pp. 4002–4008, 2008, doi: 10.1109/IJCNN.2008.4634373.

[8] M. T. Vendel, "n i t e l l Bu," no. May, 2014.

[9] X. Q. Chen, Y. Q. Chen, and J. G. Chase, "Mobiles Robots - Past Present and Future," *Mob. Robot. - State Art Land, Sea, Air, Collab. Mission.*, no. May 2014, 2009, doi: 10.5772/6986.

[10] L. Fryer, D. Coniam, R. Carpenter, and D. Lăpușneanu, "Bots for language learning now: current and future directions," *Lang. Learn. Technol.*, vol. 24, no. 2, pp. 8–22, 2020, [Online]. Available: http://hdl.handle.net/10125/44719.

[11] B. C. Bc, "The Artificial Intelligence Chronicle – Open to feedback Key events in the history of artificial intelligence," no. 1456.

[12] K. F. Nielsen, M. D. Lind, L. Gerward, and N. Thorup, "Crystal growth of ttf-tcnq by solute diffusion in earth-orbit," *Phys. Scr.*, vol. 47, no. 4, pp. 596–598, 1993, doi: 10.1088/0031-8949/47/4/021.

[13] "By."

[14] P. Wang, "On Defining Artificial Intelligence," *J. Artif. Gen. Intell.*, vol. 10, no. 2, pp. 1–37, 2019, doi: 10.2478/jagi-2019-0002.

[15] الشعراني. ف إ and الوزير. ج. غ, *No Title* □ □□□□□□□ □□□□□□□□ □□□□□□□□□ □□□□□□□ □□□□□□ □□□□□□□□□, vol. 1999, no. December. 2006.

[16] C. Bartneck, C. Lütge, A. Wagner, and S. Welsh, "Application Areas of AI," *SpringerBriefs in Ethics*, pp. 71–81, 2021, doi: 10.1007/978-3-030-51110-4_9.

[17] V. D. Soni, "International engineering journal for research & development," *Int. Eng. J.*

Res. Dev., vol. 3, no. 1, pp. 1–9, 2018.

[18] M. Hardalov, I. Koychev, and P. Nakov, "Towards automated customer support," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 11089 LNAI, no. August, pp. 48–59, 2018, doi: 10.1007/978-3-319-99344-7_5.

[19] M. Schneider and A. Hoess, "Applications of AI in Transportation Industry," no. September, 2021.

[20] S. Sepasgozar *et al.*, *A systematic content review of artificial intelligence and the internet of things applications in smart home*, vol. 10, no. 9. 2020.

[21] D. Buhalis and I. Moldavska, *Information and Communication Technologies in Tourism 2021*. Springer International Publishing, 2021.

[22] P. Pattnayak and O. P. Jena, "Innovation on Machine Learning in Healthcare Services–An Introduction," *Mach. Learn. Healthc. Appl.*, no. April, pp. 1–15, 2021, doi: 10.1002/9781119792611.ch1.

[23] S. L. Wamba-Taguimdje, S. Fosso Wamba, J. R. Kala Kamdjoug, and C. E. Tchatchouang Wanko, "Influence of artificial intelligence (AI) on firm performance: the business value of AI-based transformation projects," *Bus. Process Manag. J.*, vol. 26, no. 7, pp. 1893–1924, 2020, doi: 10.1108/BPMJ-10-2019-0411.

[24] S. . Willy Yuberto Andrisma, "the Power of Teacher-Student Relationships in Determining Student Success," *Pembagian Harta Waris Dalam Adat Tionghoa Di Kec. Ilir Timur I Kota Palembang*, vol. 1, p. 284, 2011, [Online]. Available: https://core.ac.uk/download/pdf/11715904.pdf.

[25] S. M. Timur and M. Nur, "Challenges of Estabilishing Hospital Disaster Plan," *J. Dialog dan Penanggulangan Bencana*, vol. 4, no. 1, pp. 45–55, 2013, [Online]. Available: https://perpustakaan.bnpb.go.id/jurnal/index.php/JDPB/article/view/63.

[26] R. Heckel, "Graph transformation in a nutshell," *Electron. Notes Theor. Comput. Sci.*, vol. 148, no. 1 SPEC. ISS., pp. 187–198, 2006, doi: 10.1016/j.entcs.2005.12.018.

[27] J. Grandgirard, D. Poinsot, L. Krespi, J. P. Nénon, and A. M. Cortesero, "Costs of secondary parasitism in the facultative hyperparasitoid Pachycrepoideus dubius: Does host size matter?," *Entomol. Exp. Appl.*, vol. 103, no. 3, pp. 239–248, 2002, doi: 10.1023/A.

[28] J. Davenport, *Symbolic computation*. 1991.

[29] L. Budaghyan, I. Ivkovic, and N. Kaleyski, *Triplicate functions*, no. 1. Springer US, 2022.

[30] S. Cafieri, A. Mucherino, G. Nannicini, F. Tarissan, and L. Liberti, "8 th Cologne-Twente Workshop on Graphs and Combinatorial Optimization Ecole Polytechnique and CNAM Proceedings of the Conference," no. January, 2009.

[31] C. Crawford, "The Art Game Computer Game Design," 1982.

[32] P. Baum, “Tic-Tac-Toe Tic-Tac-Toe,” no. December 1975, 2021.

[33] A. Intelligence, “Problem Solving , Search and Control Strategies Artificial Intelligence Problem Solving , Search and Control Strategies Artificial Intelligence,” 2010.

[34] B. C, “Problem Solving as State Space Search,” *Spring*, pp. 1–34, 2003.

[35] F. G. Becker *et al.*, “No 主観的健康感を中心とした在宅高齢者における 健康関連指標に関する共分散構造分析Title,” *Syria Stud.*, vol. 7, no. 1, pp. 37–72, 2015, [Online]. Available: https://www.researchgate.net/publication/269107473_What_is_governance/link/548173090cf22525dcb61443/download%0Ahttp://www.econ.upf.edu/~reynal/Civil wars_12December2010.pdf%0Ahttps://think-asia.org/handle/11540/8282%0Ahttps://www.jstor.org/stable/41857625.

[36] V. C. Chijindu, “Search in artificial intelligence,” *Choice Rev. Online*, vol. 26, no. 04, pp. 26-2172-26–2172, 1988, doi: 10.5860/choice.26-2172.

[37] K. Kask, “Set 2: State-spaces and Uninformed Search ICS 271 Fall 2016 Kalev Kask,” 2016.

[38] S. J. Russell and P. Norvig, “Solving problems by searching,” *Artif. Intell. A Mod. approach*, pp. 64–120, 2016.

[39] S. Huang, J. Cheng, and H. Wu, “Temporal Graph Traversals: Definitions, Algorithms, and Applications,” no. June 2016, 2014, [Online]. Available: http://arxiv.org/abs/1401.1919.

[40] L. Pasteur and R. Koch, “1. Introduction 1. Introduction,” vol. 74, no. 1934, pp. 535–546, 1941.

[41] D. Structures, “Graph problems intro Administrivia.”

[42] A. Auer and H. Kaindl, “A case study of revisiting best-first vs. depth-first search,” *Front. Artif. Intell. Appl.*, vol. 110, no. May, pp. 141–145, 2004.

[43] E. State, “A . I .: Solving problems by searching.”

[44] “Intro to BFS / DFS.”

[45] I. D. Suter and Q. Li, “Artificial Intelligence Search Topics § Agents that Plan Ahead § Search Problems § Uninformed Search Methods.”

[46] J. Dillenburg, “Techniques for improving the efficiency of heuristic search,” no. September 1998, 1993, [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.8.5702&rep=rep1&type=pdf.

[47] M. Phillips, V. Narayanan, S. Aine, and M. Likhachev, “Efficient search with an ensemble of heuristics,” *IJCAI Int. Jt. Conf. Artif. Intell.*, vol. 2015-Janua, no. July, pp. 784–791, 2015.

[48] C. V Patton, D. S. Sawicki, and J. J. Clark, *Basic Methods of Policy Analysis and*

Planning Third Edition. 2016.

[49] J. Recker, M. Indulska, P. F. Green, A. Burton-Jones, and R. Weber, "Information systems as representations: A review of the theory and evidence," *J. Assoc. Inf. Syst.*, vol. 20, no. 6, pp. 735–786, 2019, doi: 10.17705/1jais.00550.

[50] E. Roventa and T. Spircu, "Knowledge representation," *Stud. Fuzziness Soft Comput.*, vol. 227, pp. 13–30, 2009, doi: 10.1007/978-3-540-77463-1_2.

[51] S. D. Brunn and M. Dodge, "What is where? The role of map representations and mapping practices in advancing scholarship," *Mapp. Across Acad.*, pp. 1–22, 2017, doi: 10.1007/978-94-024-1011-2_1.

[52] T. Ten Berge and R. Van Hezewijk, "Procedural and Declarative Knowledge: An Evolutionary Perspective," *Theory Psychol.*, vol. 9, no. 5, pp. 605–624, 1999, doi: 10.1177/0959354399095002.

[53] C. Jiamu, "The great importance of the distinction between declarative and procedural knowledge," *Análise Psicológica*, vol. 19, no. 4, pp. 559–566, 2012, doi: 10.14417/ap.387.

[54] T. Johnnie, "Difference B W Declarative And Procedural Knowledge."

[55] J. Dix, "The logic programming paradigm," *AI Commun.*, vol. 11, no. 2, pp. 123–131, 1998, doi: 10.1007/978-1-84882-914-5_12.

[56] S. E. E. Profile, "The Logic Programming Paradigm J urgen Dix INFORMATIK Universitat Koblenz-Landau," no. November, 2013.

[57] G. N. Product and R. Two, "Chapter 1 : Introduction Chapter 1 : Introduction," *Fluid Mech.*, no. October, pp. 1–16, 1966.

[58] J. Wu and S. Shang, "Managing uncertainty in ai-enabled decision making and achieving sustainability," *Sustain.*, vol. 12, no. 21, pp. 1–17, 2020, doi: 10.3390/su12218758.

[59] R. Triska, "Artificial Intelligence , classification theory and the uncertainty reduction process," no. January 2007, pp. 479–483, 2014.

[60] A. Saffiotti, "An Al view of the treatment of uncertainty," *Knowl. Eng. Rev.*, vol. 2, no. 2, pp. 75–97, 1987, doi: 10.1017/S0269888900000795.

[61] T. Denœux, D. Dubois, and H. Prade, "Representations of Uncertainty in Artificial Intelligence: Probability and Possibility," *A Guid. Tour Artif. Intell. Res.*, pp. 69–117, 2020, doi: 10.1007/978-3-030-06164-7_3.

[62] M. F. Triola, "Bayes theorem: Fully informed rational estimates of diagnostic probabilities.," *J. Am. Dent. Assoc.*, vol. 141, no. 6, pp. 658–659, 2010.

[63] J. Halliwell, J. Keppens, and Q. Shen, "Linguistic Bayesian Networks for reasoning with subjective probabilities in forensic statistics," *Proc. Int. Conf. Artif. Intell. Law*, no. January, pp. 42–50, 2003, doi: 10.1145/1047788.1047795.

[64] A. Riadi, “Penerapan Metode Certainty Factor Untuk Sistem Pakar Diagnosa Penyakit Diabetes Melitus Pada Rsud Bumi Panua Kabupaten Pohuwato,” *Ilk. J. Ilm.*, vol. 9, no. 3, pp. 309–316, 2017, doi: 10.33096/ilkom.v9i3.162.309-316.

[65] D. Deslianti, “Penerapan Metode Certainty Factor Dalam Mendiagnosa Penyakit Pada Mata Manusia,” *Agustus*, vol. 3, no. 4, pp. 2655–755, 2020, [Online]. Available: https://jurnal.ikhafi.or.id/index.php/jukomika/456.

[66] C. Huang, X. Mi, and B. Kang, “Basic probability assignment to probability distribution function based on the Shapley value approach,” *Int. J. Intell. Syst.*, vol. 36, no. 8, pp. 4210–4236, 2021, doi: 10.1002/int.22456.

[67] R. Jiroušek and P. P. Shenoy, “A new definition of entropy of belief functions in the Dempster–Shafer theory,” *Int. J. Approx. Reason.*, vol. 92, no. November, pp. 49–65, 2018, doi: 10.1016/j.ijar.2017.10.010.

[68] M. Hafeez, “Application of Dempster Shafer Theory to Assess the Status of Sealed Fire in a Coal Mine,” no. May, 2011.

[69] M. Fields, “CHAPTER 2 Expert System Development Life Cycle (ESDLC),” 1996.

[70] G. Menges and N. Hoevelmanns, “Expert Systems.,” *Kunststoffe - Ger. Plast.*, vol. 77, no. 6, pp. 8–10, 1987.

[71] E. Systems, “161 - EXPERT SYSTEMS: AN OVERVIEW Felisa Verdejo Facultad de Informática, Universidad del País Vasco, Spain.,” pp. 161–169.

[72] A. Móviles, P. La, G. D. E. Fallas, S. A. Gutiérrez, and J. W. Branch, “a Comparison Between Expert Systems and Autonomic Computing Plus Mobile Agent Approaches for Fault Management . Sistemas Expertos Y Computación Autónoma Más,” no. November 2016, pp. 173–180, 2011.

[73] C. Two, R. E. Systems, and P. By, “Chapter Two : Rule-Based Expert Systems By : Dr Muhanad Tahrir Younis.”

[74] S. Mehdi, *Expert systems development: some problems, motives and issues in an exploratory study*. 1993.

[75] A. S. Shaukat and B. S. E. E, “by IN Submitted to the Graduate Faculty of Texas Tech University in Partial Fulfillment of the Requirements for the Degree of MASTER OF SCIENCE IN Approved Accepted August , 1988,” *System*, 1988.

[76] E. C. Ogu and Y. A. Adekunle, “Basic Concepts of Expert System Shells and an Efficient Model for Knowledge Acquisition,” *Int. J. Sci. Res.*, vol. 2, no. 4, pp. 554–559, 2013, [Online]. Available: https://www.ijsr.net/search_index_results_paperid.php?id=IJSRON2013922.

[77] S. G. Tzafestas, A. Kokkinaki, and K. P. Valavanis, “An Overview of Expert Systems,” pp. 1–2, 1993.

[78] P. Boucher, *Artificial intelligence: How does it work, why does it matter, and what can we do about it?*, no. June. 2020.

[79] H. Tan, “A brief history and technical review of the expert system research,” *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 242, no. 1, 2017, doi: 10.1088/1757-899X/242/1/012111.

[80] C. Rattanaprateep and S. Chittayasothorn, “A Frame-based Object-Relational Database Expert System Architecture and Implementation,” *Proc. 5th WSEAS Int. Conf. Artif. Intell. Knowl. Eng. Data Bases, Madrid, Spain, Febr. 15-17.*, vol. 2006, pp. 327–332, 2006.

[81] W. Supartini and H. Hindarto, “Sistem Pakar Berbasis Web Dengan Metode Forward Chaining Dalam Mendiagnosa Dini Penyakit Tuberkulosis Di Jawa Timur,” *Kinetik*, vol. 1, no. 3, p. 147, 2016, doi: 10.22219/kinetik.v1i3.123.

[82] J. J. Eckert, F. M. Santiciolli, R. Y. Yamashita, F. C. Corrêa, L. C. A. Silva, and F. G. Dedini, “Fuzzy gear shifting control optimisation to improve vehicle performance, fuel consumption and engine emissions,” *IET Control Theory Appl.*, vol. 13, no. 16, pp. 2658–2669, 2019, doi: 10.1049/iet-cta.2018.6272.

[83] J. D. Barrett, “Advanced Fuzzy Logic Technologies in Industrial Applications,” *Technometrics*, vol. 49, no. 4, pp. 494–495, 2007, doi: 10.1198/tech.2007.s689.

[84] F. . Guély and F. G. S. Chevrie, “Cahier technique n° 191: Fuzzy logic,” *Cah. Tech.*, p. 32, 1998.

[85] R. Article, “Fuzzy Logic in Process Safety Modeling of,” no. September, 2014.

[86] M. R. Sarmasti Emami, “Fuzzy Logic Applications In Chemical Processes,” *J. Math. Comput. Sci.*, vol. 01, no. 04, pp. 339–348, 2010, doi: 10.22436/jmcs.001.04.11.

[87] M. Ilham Rofiqi and H. Hindarto, “Analisis Kecanduan Game Player Unknown’s Battlegrounds (PUBG) Mobile dengan Menggunakan Logika Fuzzy,” *J. Inform. Polinema*, vol. 7, no. 2, pp. 97–102, 2021, doi: 10.33795/jip.v7i2.327.

[88] Br, “Science of the Brain,” *Neuroscience*, p. 56, 2003.

[89] M. C. Nwadiugwu, “Neural Networks, Artificial Intelligence and the Computational Brain,” 2020, [Online]. Available: http://arxiv.org/abs/2101.08635.

[90] L. D. Jackel, R. E. Howard, H. P. Graf, B. Straughn, and J. S. Denker, “Artificial Neural Networks for Computing.,” *J. Vac. Sci. Technol. B Microelectron. Nanom. Struct.*, vol. 4, no. 1, pp. 61–63, 1986, doi: 10.1116/1.583351.

[91] I. A. Basheer and M. Hajmeer, “Artificial neural networks: Fundamentals, computing, design, and application,” *J. Microbiol. Methods*, vol. 43, no. 1, pp. 3–31, 2000, doi: 10.1016/S0167-7012(00)00201-3.

[92] S. Akthar, “A Study on Neural Network Architectures,” vol. 7, no. 9, pp. 1–7, 2016, [Online]. Available: www.iiste.org.

[93] J. M. Zurada, A. Malinowski, and P. Przestrzelski, “Modified Hebbian learning rule for single layer learning,” *Proc. - IEEE Int. Symp. Circuits Syst.*, vol. 4, no. August 2018, pp. 2407–2410, 1993, doi: 10.1109/iscas.1993.693175.

[94] B. Widrow and M. A. Lehr, “30 Years of Adaptive Neural Networks: Perceptron, Madaline, and Backpropagation,” *Proc. IEEE*, vol. 78, no. 9, pp. 1415–1442, 1990, doi: 10.1109/5.58323.

[95] A. E. Fall and D. Naumetc, “Artificial Neural Networks in plane control,” 2016.

[96] P. Marius-Constantin, V. E. Balas, L. Perescu-Popescu, and N. Mastorakis, “Multilayer perceptron and neural networks,” *WSEAS Trans. Circuits Syst.*, vol. 8, no. 7, pp. 579–588, 2009.

[97] Hindarto, I. Anshory, and A. Efiyanti, “Feature Extraction of Heart Signals using Fast Fourier Transform,” *Proceeding 1st IBSC Towar. Ext. Use Basic Sci. Enhancing Heal. Environ. Energy Biotechnol.*, vol. 10, no. 2, pp. 165–167, 2016.

[98] M. O. Okwu and L. K. Tartibu, “Genetic Algorithm,” *Stud. Comput. Intell.*, vol. 927, pp. 125–132, 2021, doi: 10.1007/978-3-030-61111-8_13.

[99] R. N. Mir and A. A. Khan, “Optimization of Constrained Function Using Genetic Algorithm,” no. Deb 2014, pp. 11–15, 2017.

[100] D. Samanta, “Solving Optimization Problems Introduction to Solving Optimization Problems Today ’ s Topics,” pp. 1–22, 2014.

[101] M. Tabassum, “a Genetic Algorithm Analysis Towards Optimization Solutions,” *Int. J. Digit. Inf. Wirel. Commun.*, vol. 4, no. 1, pp. 124–142, 2014, doi: 10.17781/p001091.

[102] M. Gutowski, *Biology, Physics, Small Worlds and Genetic Algorithms*, no. January 2005. 2005.

BIODATA PENULIS

Dr.Hindarto, S.Kom, MT. dilahirkan di Surabaya, 30 Juli 1973. Pada tahun 1995, penulis mendapatkan gelar Diploma dari Politeknik Negeri Surabaya dan melanjutkan jenjang Sarjana di prodi Informatika FakultasTeknik Umsida. Penulis melanjutkan magister Teknik Elektro ITS dengan program beasiswa dari DIKTI. Tahun 2007, penulis secara resmi mendakatkan gelar M.T. Penulis melanjutkan doctoral di Jaringan Cerdas Multimedia Teknik Elektro ITS dengan program beasiswa dari DIKTI. Tahun 2016, penulis secara resmi mendakatkan gelar Dr. Penulis mengawali karirnya sebagai Dosen di prodi Informatika Universitas Muhaammdiyah Sidoarjo pada Tahun 2004. Beberapa penelitian yang pernah dilakukan oleh penulis adalah tentang Deteksi Sinyal Jantung dan Deteksi Sinyal EEG.

Ir. Sumarno, MM. Dilahirkan di Surabaya, 27 Mei 1961. Pada tahun 1991, penulis mendapatkan gelar Sarjana teknik elektro dari Universitas Muhammadiyah Suranaya dan melanjutkan jenjang Sarjana S2 Di Universitas Muhammadiyah Malang lulus tahun 2008, sejak tahun 1993, terdaftar sebagai Dosen tetap di Universitas Muhammadiyah Sidoarjo, mengampu mata kuliah Sistem Informasi, Sistem Informasi Manajemen, Jaringan komputer dan teori Bahasa dan Automata.

Mochamad Alfan Rosid, S.Kom., M.Kom. lahir di Sidoarjo, 25 April 1986. Lulus Sarjana Komputer Universitas Muhammadiyah Sidoarjo tahun 2010. Penulis melanjutkan studi S2 di Prodi Teknologi Informasi Program Pascasarjana Sekolah Tinggi Teknik Surabaya lulus tahun 2014 dengan mendapatkan gelar M.Kom. Saat ini penulis sedang studi lanjut S3 di program studi Ilmu Komputer ITS. Penulis mengawali karirnya sebagai Dosen di prodi Teknik Informatika Universitas Muhammadiyah Sidoarjo. Penulis juga aktif terlibat dalam kegiatan penelitian dan pengabdian kepada masyarakat. Penelitian yang pernah dilakukan oleh penulis adalah tentang sistem informasi berbasis, basis data dan sistem pengambilan keputusan. Selain pendidikan dan pengajaran penulis juga terlibat dalam kegiatan penelitian dan pengabdian kepada masyarakat baik didanai oleh Ristekdikti maupun dana mandiri. Penulis juga aktif dalam mengikuti kegiatan-kegiatan penunjang akademik seperti seminar, workshop/ lokakarya, pelatihan serta pembimbingan tugas akhir dan kegiatan akademik

UMSIDA PRESS
Universitas Muhammadiyah Sidoarjo
Jl. Mojopahit No. 666B
Sidoarjo, Jawa Timur

ISBN 978-623-464-034-2 (PDF)