BAB II

LANDASAN TEORI

2. 1 Tinjauan Pustaka

Berikut adalah beberapa penelitian terdahulu yang terkait dengan topic penelitian Multicriteria Decision Making, yang dijadikan tinjauan pustaka dan referensi dalam penelitian yang dilakukan oleh penulis :

1. *Simple Additive Weigting approach to personel selection problem* (Alireza Afshari, (2010))

   Dalam penelitian ini peneliti menggunakan metode SAW untuk melakukan seleksi personil dalam menentukan personil yang cocok. Dimana terdapat beberapa kriteria yang dijadikan penilaian untuk personil seperti kemampuan hard skill, soft skill dan attribute penilaian lainnya. Dan menggunakan penilaian dari opini pakar dalam menilai personil untuk menjadi penilaian setiap attributnya.

2. *Multi attribute Decision Making Approach by TOPSIS Technique* (P.K Parida, (2013))

   Dalam penelitian ini peneliti melakukan seleksi projek industry. Dimana terdapat penilaian attribute teknikal dan finansial. Dan menghasilkan alternative projek yang tepat untuk dijalankan.

3. Usulan Prioritas Peringkat dalam pemilihan *supplier* Produk Yamato dengan metode Promethee Studi Kasus PT. Chitose Mfg (Dzikri Arbawan Rahmatullah, (2013))

   Dalam penelitian ini peneliti melakukan Pemilihan supplier untuk kondisi ideal yakni memilih salah satu supplier terbaik menggunakan metode promethee.

4. *DSS using AHP in Selection of Lecturer* (Adriyendi, (2013))

   Dalam penelitian ini peneliti melakukan seleksi pengajar untuk universitas STAIN Batsangkar menggunakan metode AHP. Dimana data dikoleksi dari observasi dan interview oleh pusat data administrasi akademik.

5. Penerapan metode Weighted Product Model untuk seleksi calon karyawan (Lestari, (2013))

   Dalam penelitian ini peneliti melakukan seleksi karyawan untuk sebuah perusahaan menggunakan metode WPM. Dilakukan beberapa tahapan SPK menggunakan WPM untuk penilaian test awal , test kedua dan test akhir.

6. *A VIKOR method for solving personel training selection problem* (El.Santawy, (2012))

   Dalam penelitian ini peneliti melakukan seleksi personel menggunakan metode vikor.

7. *Trusted Network Selection using SAW and TOPSIS algorithm for heterogeneous wireless network* (network, (2011))

   Dalam penelitian ini peneliti melakukan seleksi visitor network untuk koneksi pada mobile terminal dimana terdapat network yang berbeda-beda seperti wifi,

wimax , wlan, wpan dsb. Beberapa kriteria seperti kekuatan signal , signal link dsb. Dilakukan perbandingan antara SAW dan TOPSIS.

8. MADM *Tool*: Aplikasi uji sensitifitas untuk model MADM menggunakan metode SAW dan TOPSIS (S, (2010))

   Dalam penelitian ini peneliti menggunakan metode SAW dan TOPSIS dalam melakukan uji sensitifitas untuk kasus MADM tertentu. Permasalahannya adalah untuk menyelesaikan sebuah kasus MADM tertentu mungkin akan sulit untuk menentukan metode mana yang paling relevan antara metode SAW dan TOPSIS. Dilakukan sebuah uji sensitivitas untuk menentukan metode yang paling tepat antara SAW atau TOPSIS dalam penyelesaian kasus tersebut. Dalam hal ini penyelesaian masalah Multi Attributes Decision Making (MADM) yang dilakukan melalui proses Uji Sensitivitas terlebih dahulu kemudian dengan metode antara SAW atau TOPSIS ini bertujuan untuk membantu setiap orang dalam pengambilan keputusan berdasarkan nilai alternative terbaik.

9. *Comparison of multi criteria decision making method from the maintenance alternative selection perspective* (Jureen Thor, (2013))

   Dalam penelitian ini peneliti melakukan seleksi alternative perawatan industry manufaktur. Dilakukan perbandingan antara AHP , ELECTRE , SAW dan TOPSIS.

10. Multi-Criteria Decision Making: An overview of different selection problems and methods. (Gavade, (2014))

Dalam penelitian ini dimana melakukan penelaahan antara beberapa metode MCDM dengan berberapa study kasus yang cocok dengan pemilihan metode MCDM yang diteliti. Diantaranya metode AHP, TOPSIS, VIKOR dan PROMETHEE.

11. Comparative Study of Different Multi-criteria Decision-making Methods (S V. G., (2013))

    Dalam penelitian ini dimana melakukan perbandingan MCDM dengan study kasus pemilihan rumah. Diantaranya metode AHP, SAW, WPM, PROMETHEE, TOPSIS dan VIKOR.

12. *An Analysis of Multi-Criteria Decision Making Methods* (Hester, Methods, An Analysis of Multi-Criteria Decision Making, (2013))

    Dalam penelitian ini dilakukan perbandingan beberapa metode MCDM secara umum. Diantaranya AHP, ELECTRE, TOPSIS, SAW, SMART, PROMETHEE dan metode lainnya.

2. 2 Multicriteria Decision Making

Ketidakpastian goal/tujuan dalam pengambilan keputusan merupakan manifestasi penting dari ketidakpastian yang berkaitan dengan karakter multikriteria decision making (Pedrycz, Fuzzy Multicriteria Decision Making, Model, Method and Application, 2011) dari banyak masalah yang dihadapi dalam proyek, perencanaan, operasi, dan kontrol sistem yang kompleks. Beberapa profesional di lapangan dari pengambilan keputusan dan analisis sistem (misalnya, Lyapunov, 1972) setuju bahwa,

dari sudut pandang umum, jenis ketidakpastian adalah yang paling sulit untuk mengobati dan mengatasi karena "kita tidak tahu apa yang kita inginkan". Pada kenyataannya, jenis ketidakpastian tidak dapat efektif ditangkap atas dasar menerapkan model formal dan metode, karena kadang-kadang sumber informasi yang unik adalah individu yang membuat keputusan.

Multikriteria pengambilan keputusan (MCDM) (Pedrycz, Fuzzy Multicriteria Decision Making, Model, Method and Application, 2011) terkait dengan pengambilan keputusan di hadapan beberapa dan kriteria yang saling bertentangan. Multikriteria pengambilan keputusan masalah dapat berkisar dari masalah keputusan sehari-hari, seperti pembelian mobil, untuk orang-orang yang mempengaruhi seluruh negara, seperti dalam bijaksana penggunaan uang untuk memelihara keamanan nasional (Lu et al., 2007). Namun, bahkan dengan keragaman yang ada ini, semua multikriteria masalah pengambilan keputusan share karakteristik umum sebagai berikut (Hwang dan Yoon, 1981):

- multiple kriteria: setiap masalah memiliki beberapa kriteria, yang dapat berupa tujuan atau atribut;
- Kriteria yang bertentangan: beberapa kriteria yang dimana terdapat konflik satu sama lain;
- unit yang dapat dibandingkan: kriteria mungkin memiliki unit yang berbeda dari pengukuran;

- desain / pilihan: solusi untuk masalah multikriteria pengambilan keputusan yang baik untuk merancang alternatif terbaik (s) atau untuk memilih yang terbaik di antara alternatif yang terbatas yang ditentukan sebelumnya.

Melalui penjelasan di atas, kita membedakan dua jenis kriteria: tujuan dan atribut. Sedemikian rupa, multikriteria masalah pengambilan keputusan dapat diklasifikasikan menjadi dua Kelas:

- *multiobjective* decision-making;
- *multiattribute* decision-making.

Perbedaan utama antara dua kelas ini adalah bahwa yang pertama berkonsentrasi pada ruang keputusan secara terus menerus dan fokus kedua pada masalah dengan ruang keputusan diskrit. Untuk menjelaskan lebih lanjut, beberapa konsep dasar dan terminologi yang diberikan di bawah. Dimana sejalan dengan notasi yang disajikan dalam literatur (Hwang dan Masud, 1979; Hwang dan Yoon, 1981; Lu et al., 2007).

- Kriteria , membentuk standar penilaian atau aturan untuk menguji penerimaan. Dalam literatur pengambilan keputusan multikriteria, dimana menunjukkan tujuan dan / atau atribut.
- Objektif , adalah refleksi dari keinginan DM (Decision Making) dan menunjukkan arah yang DM ingin berkonsentrasi. masalah pengambilan keputusan Multiobjective, sebagai hasilnya, melibatkan desain alternatif yang mengoptimalkan atau setidaknya memenuhi tujuan DM.

- Goal, merupakan entitas yang diinginkan oleh DM dan dinyatakan dalam keadaan tertentu dalam ruang dan waktu. Jadi, sementara Goal memberikan arah yang diinginkan, goal memberikan yang diinginkan (atau target) tingkat untuk dicapai.
- Atribut, adalah karakteristik, kualitas, atau parameter kinerja alternatif. Multiattribute masalah pengambilan keputusan melibatkan pemilihan "terbaik" alternatif dari kolam alternatif terpilih dijelaskan dalam hal atribut mereka.

Multiobjective pengambilan keputusan dikenal sebagai jenis pengambilan keputusan multikriteria terus menerus dan karakteristik utamanya adalah bahwa DM perlu untuk mencapai beberapa tujuan sementara tujuan ini noncommensurable dan bertentangan satu sama lain. Sebuah multiobjective pengambilan keputusan Model mencakup vektor variabel keputusan, fungsi tujuan yang menggambarkan tujuan, dan kendala. DM upaya untuk memaksimalkan atau meminimalkan fungsi objektif.

Multiattribute pengambilan keputusan terkait dengan membuat keputusan preferensi (yaitu, perbandingan, pilihan, prioritas, dan / atau memesan) atas alternatif yang tersedia yang ditandai oleh beberapa, biasanya saling bertentangan, atribut. Keunikan utama pengambilan keputusan multiattribute masalah adalah bahwa biasanya ada sejumlah alternatif yang telah ditentukan, yang berhubungan dengan tingkat pencapaian atribut. Berdasarkan atribut, final keputusan dibuat.

Akhirnya, kita harus membahas secara rinci konsep alternatif. Bagaimana menghasilkan alternative adalah bagian penting dari proses multiobjective dan

multiattribute model pengambilan keputusan bangunan (Lu et al., 2007). Di hampir semua multiobjective pengambilan keputusan model, alternative dapat dihasilkan secara otomatis oleh model. Dalam kasus multiattribute pengambilan keputusan model, namun, perlu untuk menghasilkan alternatif secara manual. Kadang, esensi masalah mendefinisikan jumlah alternatif. Namun, secara umum, bagaimana dan kapan harus berhenti menghasilkan alternatif menjadi isu yang sangat penting. Menghasilkan alternatif signifikan tergantung pada ketersediaan dan biaya informasi, dan juga membutuhkan ketergantungan pada keahlian dalam masalah daerah. Alternatif dapat dihasilkan dengan menggunakan heuristik juga, dan mereka bisa datang dari baik perseorangan maupun kelompok.

2.2. 1 Metode SAW

Churchman dan Ackoff (1954) pertama kali menggunakan metode SAW (Gwo-Hshiung Tzeng, (2011)) (Simple Additive Weighting) untuk mengatasi portofolio masalah pilihan. Metode SAW adalah mungkin yang paling dikenal dan banyak digunakan Metode untuk beberapa atribut pengambilan keputusan MADM. Karena kesederhanaannya, SAW adalah metode yang paling populer di masalah MADM dan alternatif terbaik bisa diturunkan dengan persamaan berikut:

$$A^* = \{ u_i(x) \mid \overset{max}{i} u_i(x) \mid i = 1,2 \dots n \} \qquad (1)$$

Atau gap dari alternative dapat diperbaiki untuk membangun alternative terbaik A^* untuk mencapai tingkat aspirasi/yang diinginkan pada setiap kriteria.

$$u_i(x) = \sum_{j=1}^{n} w_j r_{ij}(x), \qquad (2)$$

Dimana $u_i(x)$ merupakan pengunaan dari ith alternative dan and $= 1,2 \ldots n$; w_j merupakan bobot dari j criterion ; $r_{ij}(x)$ merupakan normalisasi rating yang disukai dari i th alternative dengan bergantung pada j th kriteria untuk seluruh unit yang sepadan ; dan seluruh kriteria di asumsikan secara independen dengan tambahan rating normalisasi yang disukai $\left(r_{ij}(x)\right)$ dari i alternative dengan bergantung pada j kriteria dapat didefinisikan dengan :

1. Bentuk 1

   Untuk kriteria benefit (lebih besar lebih baik) , $r_{ij}(x) = x_{ij}/x_j^*$, dimana $x_j^* = max_i x_{ij}$ atau dapat juga x_j^* menjadi tingkat aspirasi/yang diinginkan , dan , dan jelas $0 \leq r_{ij}(x) \leq 1$

   Untuk kriteria cost (lebih kecil lebih baik) , $r_{ij}(x) = \left(1/x_{ij}\right)/\left(1/x_j^*\right) = \left(min_i x_{ij}\right)/\left(x_{ij}\right)$ atau dapat juga x_j^* menjadi tingkat aspirasi/yang diinginkan

2. Bentuk 2

   Untuk kriteria benefit (lebih besar lebih baik) , $r_{ij} = \left(x_{ij} - x_j^-\right)/\left(x_j^* - x_j^-\right)$ dimana $x_j^* = max_i x_{ij}$ dan $x_j^- = min_i x_{ij}$ atau dapat juga x_j^* menjadi tingkat aspirasi/yang diinginkan dan x_j^- merupakan tingkat terendah.

   Untuk kriteria cost (lebih kecil lebih baik) , $r_{ij} = \left(x_j^- - x_{ij}\right)/\left(x_j^- - x_j^*\right)$

   Dimana , sintesisasi performa adalah

$$p_i = \sum_{j=1}^{m} w_j r_{ij} \qquad (3)$$

2.2. 2 Metode TOPSIS

Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS) (Gwo-Hshiung Tzeng, (2011)) adalah metode analisis keputusan multi kriteria, yang pada awalnya dikembangkan oleh Hwang dan Yoon pada tahun 1981 dengan perkembangan lebih lanjut oleh Yoon tahun 1987, dan Hwang, lai dan Liu pada tahun 1993. TOPSIS didasarkan pada konsep bahwa alternatif yang dipilih harus memiliki jarak geometris terpendek dari solusi ideal positif dan jarak geometris terpanjang dari solusi ideal negatif. Ini adalah metode agregasi kompensasi yang membandingkan satu set alternatif dengan mengidentifikasi bobot untuk setiap kriteria, normalisasi skor untuk setiap kriteria dan menghitung jarak geometris antara masing-masing alternatif dan alternatif yang ideal, yang merupakan nilai terbaik di setiap kriteria. Asumsi TOPSIS adalah bahwa kriteria yang monoton meningkat atau menurun. Normalisasi biasanya diperlukan sebagai parameter atau kriteria sering dimensi ganjil di masalah multi-kriteria. metode kompensasi seperti TOPSIS memungkinkan trade-off antara kriteria, di mana hasil yang buruk dalam satu kriteria dapat dinegasikan oleh hasil yang baik dalam kriteria lain. Ini memberikan bentuk yang lebih realistis pemodelan dari metode non-kompensasi, yang menyertakan atau mengecualikan solusi alternatif berdasarkan pada hard cut-off.

Diberikan sekumpulan alternative , $A = \{ A_k \mid k = 1, \dots n\}$ dan sekumpulan kriteria $C = \{ C_j \mid j = 1, \dots m\}$, dimana $X = \{ X_{kj} \mid k = 1, \dots n ; j = 1, \dots m \}$ dimana merupakan sekumpulan dari rating performa dan $w = \{ w_j \mid j = 1, \dots m\}$ merupakan sekumpulan bobot , dan table informasi I = (A ,C,X,W) dapat direpresentasikan pada table 2.1 berikut

Table 2.1 Matrik Alternative

$Alternatives$	C_1	C_2	$\cdots$	C_m
A_1	x_{11}	x_{12}		x_{1m}
A_2	x_{21}	x_{22}		x_{2m}
$\vdots$	$\vdots$	$\vdots$		$\vdots$
A_n	x_{n1}	x_{n2}		x_{nm}
W	w_1	w_2		w_m

Bentuk 1

$$r_{kj}(x) = \frac{x_{kj}}{\sqrt{\sum_{k=1}^{n} x_{kj}^2}} , k = 1, \dots n; j = 1, \dots m. \quad (1)$$

Bentuk 2

Untuk kriteria benefit (lebih besar lebih baik) , $r_{kj}(x) = (x_{kj} - x_j^-)/(x_j^* - x_j^-)$, dimana $x_j^* = max_k\, x_{kj}$ dan $x_j^- = min_k\, x_{kj}$ atau pengaturan x_j^* merupakan tingkat aspirasi/yang diinginkan dan x_j^- merupakan tingkat terendah

$$v_{kj}(x) = w_j r_{kj}(x), k = 1, \dots n; j = 1, \dots m. \quad (2)$$

Positive ideal point (PIS) dan the negative ideal point (NIS) didefinisikan sebagai berikut :

$$PIS = A^{+} = \{v_1^{+}(x), v_2^{+}(x), \dots v_j^{+}(x), \dots v_m^{+}(x)\}$$

$$= \left\{ \left(\underset{k}{Max} v_{kj}(x) | \, j \; E \; j_1 \right), \left(\underset{k}{Min} v_{kj}(x) | \, j \; E \; j_2 \right) | \, k = 1, \dots n \right\} \qquad (3)$$

$$NIS = A^{-} = \{v_1^{-}(x), v_2^{-}(x), \dots v_j^{-}(x), \dots v_m^{-}(x)\}$$

$$= \left\{ \left(\underset{k}{Min} v_{kj}(x) | \, j \; E \; j_1 \right), \left(\underset{k}{Max} v_{kj}(x) | \, j \; E \; j_2 \right) | \, k = 1, \dots n \right\} \qquad (4)$$

Dimana j_1 dan $j_2$merupakan atribut benefit dan cost

Tahap berikutnya adalah melakukan perhitungan pemisahan dari PIS dan NIS diantara alternative. Nilai pemisahan dapat diukur menggunakan jarak Euclidean dimana dirumuskan sebagai berikut :

$$D_k^{*} = \sqrt{\sum_{j=1}^{m} [\, v_{kj}(x) - v_j^{+}(x)]^2} \;, k = 1, \dots . n \qquad (5)$$

Dan

$$D_k^{-} = \sqrt{\sum_{j=1}^{m} [\, v_{kj}(x) - v_j^{-}(x)]^2} \;, k = 1, \dots . n \qquad (6)$$

Kesamaan untuk PIS dapat dirumuskan sebagai berikut :

$$C_k^{*} = D_k^{-} / (D_k^{*} + D_k^{-}) \;, k = 1, \dots n \qquad (7)$$

2.2. 3 Metode Promethee

Brans et al (1984) pertama kali yang mengira bahwa ***A*** adalah himpunan berhingga dari alternatif yang mungkin. PROMETHEE (Gwo-Hshiung Tzeng, (2011)) merupakan singkatan dari Preference Ranking Organization METHod for Enrichment of Evaluations . Sebuah *partial preorder* (PROMETHEE I) atau sebuah *preorder* lengkap (PROMETHEE II) pada ***A*** dapat pertama di tawarkan untuk pembuat keputusan. PROMETHEE III memberikan sebuah order interval yang menekankan pada pengabaian. PROMETHEE IV setuju dengan kumpulan berkelanjutan dari alternative yang mungkin.

Metode PROMETHEE meminta informasi tambahan tetapi hanya pada sedikit parameter untuk diperbaiki dan seluruhnya memiliki signifikansi ekonomi nyata. 6 tipe yang mungkin dari generalisasi kriteria yang dapat dianggap dalam metode PROMETHEE dan seperti yang ditampilkan dalam table 2.1. Setiap tipe dapat dengan mudah didefinisikan karena hanya satu atau dua parameter yang perlu diperbaiki.

1. q merupakan difference threshold. Dimana merupakan nilai terbesar dari d dibawah dimana pembuat keputusan menganggap ada ketidakpedulian
2. p merupakan preference threshold. Dimana merupakan nilai terendah dari d diatas dimana pembuat keputusan menganggap ada preferensi yang ketat
3. σ merupakan parameter yang telah diketahui langsung terhubung dengan standar deviasi dari distribusi normal

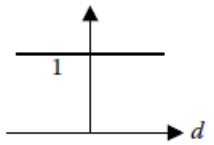
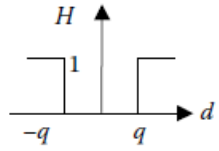
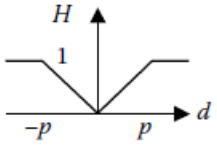
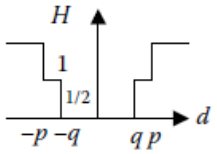
Misalkan A himpunan berhingga alternatif untuk masalah MCDM, dan mengira fungsi preferensi , f_j telah didefinisikan untuk setiap g_j untuk setiap pasangan dari

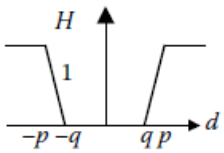
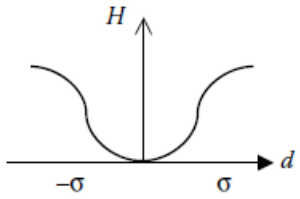
alternative $a, b \in A$ ketika $a \succ b$ dalam j kriteria.$f_j(a,b) = f_j(d_{ab|j})$ mengindikasikan bahwa derajat dari alternative lebih suka untuk alternative b (a over b) dengan jarak yang berbeda dari nilai performa $d_{ab|j} = g_j(a) - g_j(b)$ dalam j kriteria, dan $\pi(a,b)$ merupakan preferensi indek melalui seluruh kriteria yang didefinisikan oleh

$$\pi(a,b) = \sum_{j=1}^{n} w_i f_j(a,b) \qquad (1)$$

Berikut merupakan tipe metode promethee

Tabel 2. 1 Tipe Preferensi Promethee

No	Nama Tipe	Definisi Analitis	Bentuk	Parameter
1	*Type I:* Usual criterion	$H(d) = \begin{cases} 0\,, d = 0; \\ 1\,, \lvert d \rvert > 0. \end{cases}$		NA
2	*Type II:* Quasi-criterion	$H(d) = \begin{cases} 0\,, \lvert d \rvert \le q; \\ 1\,, otherwise. \end{cases}$		Q
3	*Type III:* V-sharp criterion	$H(d) = \begin{cases} \dfrac{\lvert d \rvert}{p}\,, \lvert d \rvert \le p; \\ 1\,, \lvert d \rvert > 0. \end{cases}$		P
4	*Type IV:* Level-criterion			q,p

No	Nama Tipe	Definisi Analitis	Bentuk	Parameter
		$H(d) = \begin{cases} 0\,, \lvert d\rvert \le q; \\ \frac{1}{2}, q\ < \lvert d\rvert \le p; \\ 1\,, otherwise. \end{cases}$		
5	*Type V:* Linear criterion	$H(d) = \begin{cases} 0\,, \lvert d\rvert \le q; \\ \frac{\lvert d\rvert - q}{p-q}, q\ < \lvert d\rvert \le p; \\ 1\,, otherwise. \end{cases}$		q,p
6	*Type VI:* Gaussian criterion	$H(d) = 1 - \exp\left\{-\frac{d^2}{2\sigma^2}\right\}$		Σ

The leaving flow

$$\emptyset^{+}(a) = \sum_{b \in A} \pi(a,b), \qquad (2)$$

The entering flow

$$\emptyset^{-}(a) = \sum_{b \in A} \pi(b,a), \qquad (3)$$

The net flow

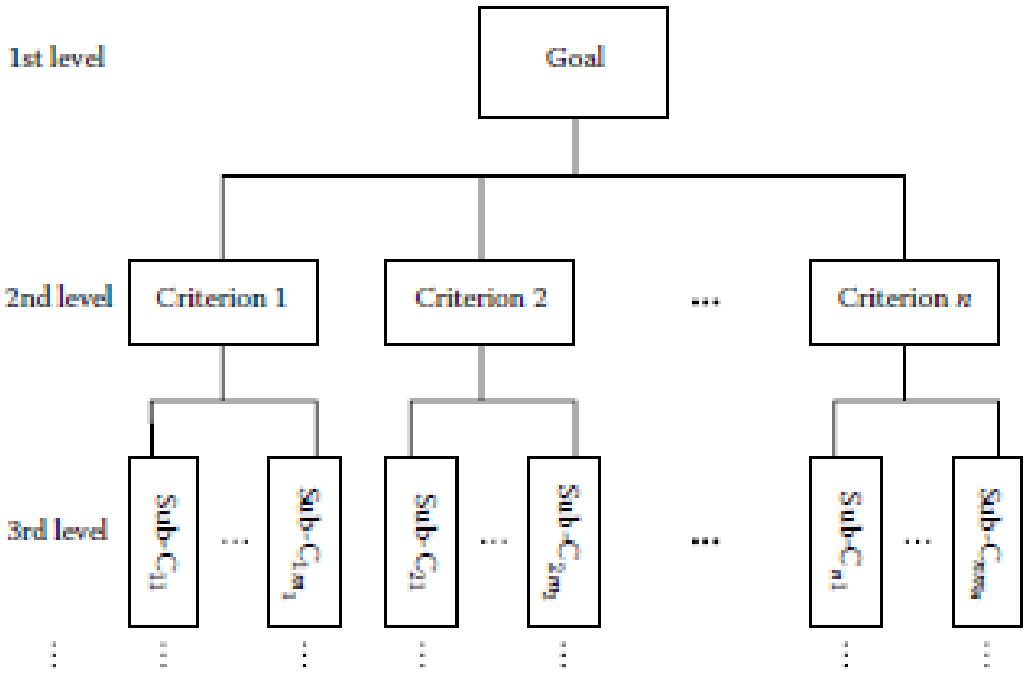
$$\emptyset(a) = \emptyset^{+}(a) - \emptyset^{-}(a). \qquad (4)$$

2.2. 4 Metode AHP

AHP (Analytic Hierarchy Proses) (Gwo-Hshiung Tzeng, (2011)) pertama kali diperkenalkan oleh Saaty (1977,1980) untuk memodelkan subjektif proses pembuatan keputusan berdasarkan pada beberapa atribut di dalam sistem hierarki. Dalam momen tersebut, telah banyak dipergunakan untuk perencanaan perusahaan, seleksi portofolio dan analisis *benefit/cost* oleh pemerintahan untuk tujuan alokasi sumber daya. Dan perlu digaris bawahi bahwa permasalahan keputusan merupakan dianggap sebagai struktur hierarki di dalam AHP. Pada tingkat pertama mengindikasikan tujuan dari permasalahan keputusan spesifik. Pada tingkat kedua, tujuan didekomposisi menjadi beberapa kriteria dan pada tingkat terbawah dapat diikuti untuk membagi menjadi sub kriteria lain.

Terdapat 4 langkah utama dari AHP yang dapat disimpulkan sebagai berikut :

Langkah 1 : menyetel sistem hierarki dengan melakukan dekomposisi permasalahan kedalam hierarki dari elemen yang saling berelasi.



Gambar 2. 1 Hierarki Struktur dari AHP

Langkah 2 : membandingkan komparasi bobot antara attribute dari elemen keputusan ke dalam bentuk reciprocal matrik.

Langkah 3 : Sintesis keputusan individu subjektif dan estimasi bobot relative

Langkah 4 : Agregasikan bobot relative dari elemen untuk menentukan alternative terbaik.

Jika kita akan membandingkan sekumpulan dari n attribute berpasangan bergantung pada bobot relative dimana attribute dilambangkan dengan $a_1, a_2, \ldots, a_n$ dan bobot dilambangkan dengan $w_1, w_2, \ldots, w_n$ kemudian dilakukan perbandingan berpasangan yang direpresentasikan sebagai berikut

$$A = \begin{bmatrix} a_{11} & \cdots & a_{1j} & \cdots & a_{1n} \\ \vdots & & \vdots & & \vdots \\ a_{i1} & \cdots & a_{ij} & \cdots & a_{in} \\ \vdots & & \vdots & & \vdots \\ a_{n1} & \cdots & a_{nj} & \cdots & a_{nn} \end{bmatrix}$$

Dimana $a_{ij} = 1/a_{ji}$ (positive reciprocal) dan $a_{ij} = a_{ik}/a_{jk}$ dicatat bahwa dalam situasi realistic w_i/w_j merupakan tidak diketahui. Dimana , permasalahan untuk ahp adalah mencari a_{ij} seperti $a_{ij} = w_i/w_j$

Dimana matrik bobot direpresentasikan sebagai berikut

$$W = \begin{array}{c} \\ w_1 \\ \vdots \\ w_i \\ \vdots \\ w_n \end{array} \begin{array}{c} \begin{array}{cccc} w_1 & \cdots & w_j & \cdots & w_n \end{array} \\ \begin{bmatrix} \frac{w_1}{w_1} & \cdots & \frac{w_1}{w_j} & \cdots & \frac{w_1}{w_n} \\ \vdots & & \vdots & & \vdots \\ \frac{w_i}{w_1} & \cdots & \frac{w_i}{w_j} & \cdots & \frac{w_i}{w_n} \\ \vdots & & \vdots & & \vdots \\ \frac{w_n}{w_1} & \cdots & \frac{w_n}{w_j} & \cdots & \frac{w_n}{w_n} \end{bmatrix} \end{array}$$

2.2. 5 Metode WPM

Metode WPM (Gwo-Hshiung Tzeng, (2011)) merupakan singkatan dari Weighted Product Method. Model ini serupa dengan SAW. Perbedaan utamanya adalah , penjumlahan di dalam model, terdapat perkalian(Miller dan Starr , 1969). Keseluruhan atau nilai komposit performa dari alternative dihitung dengan rumus berikut

$$P_i = \prod_{j=1}^{M} \left[(m_{ij})_{normal} \right]^{w_j} \qquad (1)$$

Nilai normalisasi telah dihitung seperti yang dijelaskan pada metode SAW. Setiap nilai normalisasi dari alternative dengan bergantung pada atribut seperti $(m_{ij})_{normal}$ akan ditingkatkan untuk dipangkatkan dari bobot relative dari atribut yang berkorespondensi. Alternatif dengan nilai P_i tertingi merupakan alternative terbaik.

2.2. 6 Metode Vikor

Metode VIKOR (Vlse Kriterijumska Optimizacija I Kompromisno Resenje dalam bahasa Bosnia) (Gwo-Hshiung Tzeng, (2011)) atau Multi-kriteria optimasi dan solusi kompromi dibangun untuk optimisasi multikriteria dari sistem yang kompleks. Dimana menentukan daftar rangking kompromi, solusi yang dikompromisasikan , dan stabilitas bobot interval dari solusi yang dikompromisasikan diperoleh dengan inisialisasi bobot. Metode ini berfokus pada rangking dan seleksi dari sekumpulan alternative dalam kehadiran kriteria yang bertentangan. Memperkenalkan indek rangking multikriteria berdasarkan pada pengukuran tertentu dari "*closeness*" untuk solusi "ideal" (Opricovic 1998).

Diasumsikan bahwa setiap alternative dievaluasi sesuai untuk setiap fungsi kriteria, rangking yang dikompromikan dapat dilakukan dengan membandingkan pengukuran dari kedekatan kepada alternative ideal. Pengukuran multikriteria untuk rangking yang dikompromikan dibangun dari *Lp-metric* digunakan sebagai fungsi agregasi di dalam metode pemrograman kompromisasi.

Beberapa k alternative $(k = 1, \dots n)$ dilambangkan sebagai $a_1, a_2, \dots., a_n$ untuk alternative a_k *ratingi* dari aspek j / kriteria dilambangkan dengan f_{kj} contoh f_{kj}merupakan nilai dari j th fungsi kriteria untuk alternative a_k , m merupakan jumlah kriteria $(j = 1, \dots m)$

$$L_{p,k} = \left\{\sum_{j=1}^{n}\left[w_j \left(f_j^* - f_{kj}\right)/\left(f_j^* - f_j^-\right)\right]^p\right\}^{1/p}, 1 \le p \le \infty ; k = 1,2 \dots . n \qquad (2)$$

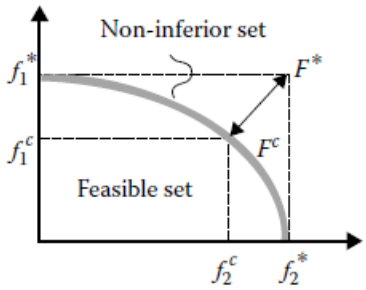
Melalui metode vikor $L_{1,k}$ dan $L_{\infty,k}$ digunakan untuk memformulasikan pengukuran rangking, solusi diperoleh dengan $min_k S_k$ merupakan maximum group utility (aturan "majority" , digambarkan sebagai kesenjangan rata-rata, ketika p = 1) dan solusi diperoleh dengan $min_k R_k$ merupakan minimum individual regret dari "opponent".

Kompromisasi solusi f^c merupaka solusi layak dimana merupakan "closest" untuk ideal f^* dan kompromisasi berarti sebuah kesepakatan dengan didirikan oleh konsesi mutual yang diilustrasikan sebagai berikut

$$\Delta f_1 = f_1^* - f_1^c \text{ and } \Delta f_2 = f_2^* - f_2^c \quad (3)$$

Algoritma kompromisasi rangking VIKOR dijelaskan dengan langkah sebagai berikut:

1. Menentukan nilai f_j^* terbaik dan f_j^- terburuk untuk seluruh fungsi kriteria $= 1,2 \dots m$. jika fungsi j th merepresentasikan *benefit* kemudian $f_j^* = max_k f_{kj}$ atau menyetel f_j^* sebagai tingkat yang diinginkan. $f_j^- = min_k f_{kj}$ atau menyetel f_j^- sebagai tingkat yang buruk.



Gambar 2. 2 Solusi ideal dan kompromisasi

2. Menghitung nilai dari S_k dan R_k , $k = 1,2 \dots n$, melalui keterkaitan

   $S_k = \sum_{j=1}^{m} w_j \left| f_j^* - f_{kj} \right| / \left| f_j^* - f_j^- \right|$, menujukan sebagai average gap

   $R_k = \underset{k}{max} \{ w_j \left| f_j^* - f_{kj} \right| / \left| f_j^* - f_j^- \right| , j = 1,2 \dots . m \}$, menunjukan sebagai maximal gap untuk perbaikan kualitas , dimana w_j merupakan bobot dari kriteria , mengekspresikan kepentingan relative

3. Mengkomputasi nilai Q_k $k = 1,2 \dots . n$ melalui keterkaitan

   $Q_k = v\,(S_k - S^*)/(S^- - S^*) + (1 - v)\,(R_k - R^*)/(R^- - R^*)$, $k = 1,2 \dots . m$

   dimana

   $S^* = \underset{k}{min} S_k$ atau $S^* = 0$ menjadi zero gap atau mencapai tingkat yang diinginkan

   $S^- = \underset{k}{max} S_k$ atau $S^- = 1$ menjadi tingkat terburuk

   $R^* = \min R_j$ atau $R^* = 0$ menjadi zero gap atau mencapai tingkat yang diinginkan

$R^- = \max R_j$ atau let $R^- = 1$ menjadi tingkat terburuk

4. Melakukan perangkingan alternative, mengurutkan dengan nilai S , R dan Q secara menurun
5. Mengusulkan sebagai solusi kompromi alternative (a') dimana dilakukan perangkingan terbaik dengan mengukur Q (minimum) jika mengikuti dua kondisi sebagai berikut :
    a. C1 "*Acceptable Advantage*"

    $$Q(a'') - Q(a') \geq DQ \qquad (4)$$

    Dimana a'' merupakan alternative dengan posisi kedua dalam daftar rangking dari Q. $DQ = \frac{1}{(J-1)}$ dan j merupakan jumlah alternative

    b. C2 "*Acceptable Stability in decision making*"

    Alternative a' harus merupakan rangking terbaik dari S atau/dan R . Merupakan kompromisasi solusi yang stabil di dalam proses pembuatan keputusan, dimana dapat berupa : "voting by majority rule" (dimana v > 0.5 diperlukan) , "by consensus" (v= 0.5) , atau "with vote" (v < 0.5). Disini v merupakan bobot untuk strategi pembuatan keputusan

Jika salah satu kondisi tidak terpenuhi , kemudian kumpulan kompomisasi solusi di tawarkan , dan dimana

- Alternative a' dan a'' jika kondisi C2 tidak terpenuhi

- Alternatif a' , a'', … , a^n jika kondisi C1 tidak terpenuhi dan $a^{(n)}$ ditentukan oleh keterkaitan $Q\left(a^{(n)} - Q(a')\right) < DQ$ untuk maksimum n (posisi dari alternative tersebut "dalam kedekatan"

Alternatif terbaik, dirangking oleh Q dimana merupakan satu dengan nilai minimum dari Q. Hasil perangkingan utama adalah daftar kompromisasi rangking dari alternative dan kompromisasi solusi dengan "advantage rate".

Perangkingan oleh VIKOR dapat dilakukan dengan nilai berbeda dari bobot kriteria, menganalisa dampak dari bobot kriteria dalam solusi kompromisasi yang ditawarkan. Metode VIKOR menentukan stabilitas interval bobot, menggunakan metodologi yang dipresentasikan Opricovic (1998). Kompromisasi solusi diperoleh dengan bobot inisial (w_j , $j = 1$, … . , m) akan diganti jika nilai dari bobot tidak berada pada stabilitas interval. Analisa dari stabilitas interval bobot untuk sebuah kriteria dilakukan untuk seluruh fungsi kriteria, dengan nilai inisial yang sama dari beban/bobot. Melalui hal ini, preferensi stabilitas dari kompromisasi solusi yang diperoleh dapat dianalisa menggunakan VIKOR program.

VIKOR merupakan alat yang sangat membantu di dalam multicriteria decision making, terutama dalam situasi dimana pembuat keputusan tidak dapat, atau tidak tahu, untuk mengekspresikan preferensi pada awal dari perancangan sistem. Kompromisasi solusi yang diperoleh dapat diterima oleh pembuat keputusan karena memberikan sebuah maksimum "group utility" (direpresentasikan dengan min S) dari "majority"

dan minimum dari penyesalan individu (direpresentasikan dengan min R) dari "opponent". Kompromisasi solusi dapat menjadi basis dari negosiasi yang melibatkan preferensi/pilihan pembuat keputusan melalui bobot kriteria.

2. 3 *Correlation Coeficient*

Correlation coefficient (Sarah Boslaugh) merupakan pengukuran dari keterkaitan antara 2 variabel, biasanya digunakan untuk menghitung derajat dari asosiasi antara 2 variabel. Korelasi digunakan ketika sebuah eksplorasi atau tingkat observasi dari riset untuk menentukan variable mana saja yang memiliki keterkaitan statistical dengan yang lainnya. Pada desain eksperimental , korelasi juga dipergunakan untuk menentukan derajat dari asosiasi antara *independent* dan *dependent* atau respon variable. Mencari dari sebuah korelasi antara 2 variabel tidak berdampak pada perubahan dalam sebuah variable yang menjadikan perubahan korespondensi di variable lainnya. Sejarah komputasi correlation coeficien sangat luas dan biasanya tanpa teoritikal atau justifikasi berdasarkan model dimana memberikan beberapa error pada *inference* yang dibuat.

2.3. 1 ***Mengukur Asosiasi***

Dunia dipenuhi dengan korelasi atau asosiasi statistic antara 2 (atau lebih) variable. Seperti keterkaitan yang berguna untuk mengkarakterisasi atau memprediksi beberapa fenomena. Sebagai contoh, pelajar ekonom tertarik dalam memahami rata-rata pergerakan bunga bank central per bulan setiap beberapa bulan / 4 bulan dalam

setahun, jika inflasi lebih besar dari 2% maka rata-rata bunga selalu naik 0.25%. pelajar melakukan evaluasi selama 10 tahun terakhir dari data dan menyimpulkan terdapat asosiasi antara kedua variable (inflasi dan rata-rata bunga meningkat) dengan sangat kuat. Hipotesa pelajar bahwa inflasi dapat menjadikan rata-rata bunga meningkat. Pelajar tersebut melakukan email kepada professor untuk melaporkan temuannya, hanya untuk mengetahui bahwa kedua variable tersebut berkorelasi, keterkaitan merupakan hal yang membuat untuk penggunaan pada bank sentral menggunakan 2% perbandingan inflasi sebagai factor pada deliberiasi.

Pelajaran penting tersebut bahwa terdapat korelasi ketika fase observasi dari riset yang ditandai sebagai asosiasi yang kuat antara 2 variabel , salah satu nya mempengaruhi lainnya. Ketergantungan sementara seperti ini merupakan contoh kandidat untuk hal keterkaitan. Pada kasus ini, pelajar dapat menentukan bahwa ketika terjadi inflasi tidak hanya memiliki hal spesifik, pergerakan rata-rata bunga ditentukan oleh komite, berdasarkan observasi pada variable. Dan hal ini berdasarkan inflasi sebagai variable independen, dan rata-rata bunga sebagai *dependen* variable. Catatan bahwa pengaturan rata-rata bunga dari konstitusi komite merupakan variable tingkat menengah, ketika pelajar dapat mengukur asosiasi antara inflasi dan pergerakan rata-rata bunga tanpa pengetahuan atau observasi sama sekali dari perilaku komite.

Catatan penting bahwa ketika mengukur dari asosiasi yang berguna untuk melakukan karakterisasi keterkaitan antara 2 variabel, dan tidak selalu merefleksikan yang terjadi dan variasi signifikan di dalam asosiasi antara kasus spesifik, Ketika kanker secara kuat diasosiasikan dengan kebiasaan merokok, dsb, beberapa orang

mendapat kanker tanpa pernah menyalakan rokok , ketika beberapa perokok berat hidup dari 100 dan mati karena hal lain.

Jika memiliki populasi yang besar, dan tidak selalu asosiasi antara 2 variabel adalah 100%, beberapa kasus mungkin terjadi. Pada kasus dari kanker, terdapat beberapa hal penyebab kanker dari merokok (seperti perokok pasif) dan juga beberapa pengukuran protektif untuk individu yang mendapat kanker dari genetic. Ketika kita dapat mengkonstruksi sebuah model dari keterkaitan antara 2 variabel , dan dapat juga mungkin untuk memperbaiki model tersebut sampai pada variasi dijelaskan.

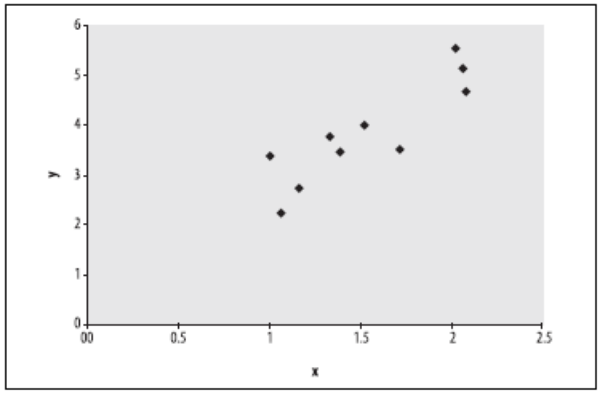
Terdapat beberapa teknik yang telah dibangun untuk menghitung asosiasi antara variable dari skala berbeda (nominal , ordinal , internal dan rasio) seperti :

- Pearson product momen (dengan kedua variable yang diukur pada interval atau skala rasio)
- Spearman rank order (dengan kedua variable yang diukur pada skala ordinal)
- Kendall (dengan kedua variable yang diukur)
- Phi (dengan kedua variable diukur pada skala nominal/dikotomi)
- Point biserial (dengan satu variable yang diukur pada skala nominal/dikotomi)

2.3. 2 Asosiasi Melalui *Scatterplot*

Salah satu cara untuk mengeksplorasi asosiasi antara variable adalah menggunakan *scatterplot*. Scatterplot merupakan sebuah grafik yang biasanya mengambil bentuk dari sebuah penjelasan variable (pada horizontal atau x-axis) di tempatkan melalui respon variable (pada vertikal , atau y-axis). Setiap anggota dari contoh berkorespondensi pada sebuah titik data dari grafik yang dideskripsikan melalui koordinat (x,y). ketika banyak variable ditempatkan , untuk seluruh pola dari keterkaitan antara penjelasan dan respon variable.keterkaitan dapat dideskripsikan sebagai kuat dan lemah.

Bentuk dari asosiasi dapat berupa linear dan non linear. Statistik dapat melihat untuk menentukan keterkaitan matematis antara kedua variable (dispesifikasikan antara model linear atau non linear) , dimana kebanyakan engineer tertarik dengan versi linear.



Gambar 2. 3 Contoh Distribusi nilai pada grafik *scatterplot*

2.3. 3 *Pearson Product Moment Correlation Coefficient*

Menggunakan scatterplot merupakan pemandu visual yang baik untuk mengevaluasi keterkaitan antara 2 variabel. Terdapat pengukuran kuantitatif yang dapat menghitung untuk menentukan kekuatan dan arah dari keterkaitan antara 2 variabel. Melalui penjelasan sederhana keterkaitan antara 2 variabel yang serupa untuk garis lurus. Jika nilai terdapat scater yang lemah melalui garis lurus pada scatterplot, variable disebutkan secara lemah berkorelasi, scatter yang lemah berarti bahwa nilai secara umum tersebar melalui nilai yang diprediksi melalui garis lurus, tetapi terdapat deviasi atau error dari nilai prediksi. Korelasi dapat berupa positif atau negative.

Correlation coefficient mengukur tendensi dari 2 variabel untuk merubah nilai bersama (seperti meningkat atau menurun). Untuk melakukan ini , jumlah dari produk kedua variable yang terstandarisasi dibagi oleh degress of freedom.

Formula untuk correlation coefficient :

$$r = \frac{\sum_{i=1}^{n}\left(\frac{x_i-\bar{x}}{s_x}\right)\left(\frac{y_i-\bar{y}}{s_y}\right)}{n-1} \quad (1)$$

Dimana melalu korelasi yang diturunkan dari jumlah produk dari standar deviasi melalui rata-rata untuk (x, y) kumpulan koordinat.

Untuk menghitung r , melalui algoritma , maka formula dijelaskan dengan langkah berikut :

- Untuk setiap (x, y) kumpulan dari koordinat , mengurangi rata-rata untuk setiap observasi x dan y

- Membagi dengan standar deviasi
- Mengalikan kedua hasil bersama
- Hasil tersebut dijumlahkan
- Jumlah tersebut dibagi dengan *degree of freedom*, n -1

R selalu pada rentang -1 atau 1 , dengan nilai yang mendekati 0 merepresentasikan keterkaitan yang lemah, dan nilai tertingi merepresentasikan keterkaitan tinggi (dapat secara kuat negative atau positif). Korelasi dengan 1 berarti kedua nilai tersebut secara sempurna berkorelasi, dan -1 berarti secara sempurna berkorelasi negative dan korelasi 0 berarti bahwa tidak terdapat keterkaitan antara kedua variable tersebut.

2.3. 4 *Spearman Rank Order Correlation Coeficient*

Kadang-kadang kita tertarik dalam menentukan keterkaitan antara kedua variable di dalam ranking dari setiap kasus pada setiap variable. Dimana merupakan kasus dimana kita tertarik pada keterkaitan ordinal. Sebagai contoh pada pengujian edukasi, dimana sulit apriori untuk mempediksi dimana pengujian dapat memproduksi distribusi dari hasil yang menjadikan identifikasi dan diskriminasi dari beberapa tipe dari tingkat kemampuan. Jika pengujian merupakan kumpulan “terlalu mudah” maka banyak pelajar yang mendapatkan “tanda penuh”, ketika pengujian “terlalu sulit” maka kebanyakan pelajar akan gagal. Pada kedua kasus tersebut terdapat variasi dalam kemampuan secara umum. Ketika membandingkan hasil melalui test, kita dapat

menjadi lebih tertarik pada rangking pelajara dalam sebuah kelas, dari pada rangking pelajar. Menggunakan rangking daripada nilai pelajar menjadikan perbandingan lebih bijak dan valid.

Spearman rank order coefficient (atau *rank correlation coefficient*) merupakan metode yang serupa dengan product moment correlation coefficient, kecuali rangking yang berkorelasi daripada nilai mentah. Sehingga banyak usaha komputasi menyertakan proses dari perbedaan antara rangking untuk item individu dalam variable yang berbeda dan kemudian mengobservasi deviasi.

Formula untuk mengkalkulasi spearman rank correlasi adalah :

$$r_s = 1 - \frac{(6 \sum d^2)}{n(n^2-1)} \qquad (1)$$

2.3. 5 *Kendal's Tau Corelation Coeficient*

Kendal tau merupakan salah satu dari beberapa pengukuran korelasi atau asosiasi. Pengukuran dari korelasi tidaklah pengujian statistic inferensi , pengukuran stastikal deksriptif yang merepresentasikan tingkat dari keterkaitan antara 2 atau lebih variable. Melalui komputasi dalam mengukur dari korelasi, praktek untuk menggunakan satu atau lebih pengujian inferential statistical dalam melakukan evaluasi satu atau lebih hipotesis yang berpusat pada correlation coeficien. Kendall tau correlation merupakan salah satu non-parametric correlation coefficient.

Dimana $x_1, \ldots, x_n$ *menjadi contoh untuk variable acak x dan dimana* $y_1, \ldots, y_n$ menjadi contoh untuk variable y acak dari ukuran yang sama n. Untuk

mendefinisikan pasangan concordant , discordant dan keduanya dijelaskan sebagai berikut :

- concordant jika ($x_i > x_j$ dan $y_i > y_j$) atau ($x_i < x_j$ dan $y_i < y_j$)
- discordant jika ($x_i > x_j$ dan $y_i < y_j$) atau ($x_i < x_j$ dan $y_i > y_j$)
- keduanya jika $x_i = x_j$ or $y_i = y_j$ (tidak dihitung)

Formula Kendal Tau correlation coefficient :

$$r = \frac{C-D}{\frac{n(n-1)}{2}} \qquad (1)$$

2. 4 *Sensitivity Analysis*

Istilah sensitivity analisis (Andrea Saltelli) diintepretasikan secara bermacam-macam pada komunitas teknis, dan pengaturan permasalahan yang berbeda-beda. Definisi yang lebih akurat dari istilah tersebut bergantung pada keluaran analisis yang dispesifikasikan.

Sensitivity analysis yang didefinisikan sebagai pengukuran local dari efek yang diberikan masukan atau keluaran. Hal ini dapat diketahui dengan melakukan penghitungan melalui pendekatan langsung atau tidak langsung , penurunan sistem seperti $Sj = \partial Y/\partial Xj$, dimana y merupakan keluaran dari yang diinginkan dan x merupakan factor masukan (rabitz , 1989; turanyi , 1990)

Pendekatan lokal tentu berharga untuk masalah yang longgar didefinisikan sebagai 'terbalik', yaitu penentuan beberapa parameter fisik tertanam menjadi model

yang kompleks dari penentuan eksperimental , diamati lebih hilir dalam model. Sebuah contoh khas adalah penentuan konstanta kinetik atau potensi mekanik kuantum dari hasil tingkat proses (Rabitz, 1989). Pendekatan ini masih cukup luas. Seperti yang dibahas dalam Saltelli (1999) sebagian besar sensitivitas analisis yang dapat ditemukan di jurnal ilmiah fisik lokal, kadang-kadang tidak tepat.

Pengaturan masalah yang berbeda untuk praktisi terlibat dalam analisis risiko (industri, keuangan, dll), pendukung keputusan, lingkungan penilaian, analisis kepatuhan terhadap peraturan, penilaian dampak, dll. Untuk ini tingkat variasi faktor input material, sebagai salah satu output yang dicari dari analisis adalah penilaian kuantitatif ketidakpastian sekitar beberapa nilai estimasi terbaik untuk Y (analisis ketidakpastian). Ini dapat dicapai dalam kasus sederhana dengan ekspresi analitik atau Taylor ekspansi, namun dicapai paling sering dan umumnya oleh Monte Carlo metode dalam hubungannya dengan berbagai strategi pengambilan sampel (Lihat Helton, 1993). Dalam konteks ini, analisis sensitivitas bertujuan, antara lain, pada pengaturan prioritas, untuk menentukan apa faktor yang paling membutuhkan tekad yang lebih baik, dan untuk mengidentifikasi keterkaitan yang lemah dari penilaian rantai (orang-orang yang menyebarkan sebagian varians dalam output). analisis sensitivitas dalam konteks ini sering dilakukan dengan menggunakan regresi teknik, seperti koefisien regresi standar.

Analisis dilakukan dengan algoritma regresi (seperti kuadrat terkecil biasa) dengan Model input dan nilai output. Itu algoritma regresi mengembalikan regresi meta-model, dimana output Y dijelaskan dalam hal kombinasi linear dari input faktor.

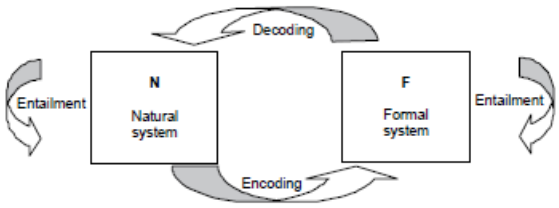
Definisi analisis sensitivitas adalah sebagai berikut: Studi tentang bagaimana ketidakpastian dalam output dari model (numerik atau sebaliknya) dapat dibagi ke berbagai sumber ketidakpastian dalam input Model (Saltelli et al., 2004). Praktek terkait adalah 'analisis ketidakpastian', yang berfokus lebih pada mengukur ketidakpastian dalam model output. Idealnya, ketidakpastian dan analisis sensitivitas harus dijalankan bersama-sama, dengan analisis ketidakpastian sebelumnya dalam praktek saat ini. Untuk definisi analisis sensitivitas untuk menjadi penggunaan, pertama kali harus dilakukan jelas apa yang dimaksudkan di sini oleh 'model', numerik atau sebaliknya, serta oleh istilah 'masukan' dan 'output' yang akan digunakan.

Pandangan dari pemodelan yang dapat membantu untuk menggambarkan peran analisis sensitivitas dalam proses ilmiah yang diambil dari karya biologi Robert Rosen (1991) (lihat juga Saltelli et al., 2000, hlm. 3-4). Pada diagram Rosen dimana memiliki 'dunia', yaitu sistem yang membentuk subjek penyelidikan. Dimana terdapat alasan untuk percaya bahwa sistem, baik alami atau buatan, diatur oleh aturan yang kita miliki ambisi untuk mengungkap, atau digunakan untuk keuntungan kita. Untuk tujuan ini kita berhipotesis satu set struktur dalam model. Misalnya, mekanisme pertumbuhan hipotesis untuk spesies yang terdapat di dunia dapat diterjemahkan ke dalam persamaan diferensial dalam model.

Intuisi Rosen adalah bahwa sementara spesies di dunia mematuhi aturan, dan persamaan diferensial dalam model memiliki 'Aturan' juga, apakah formal atau matematika, tidak ada 'aturan' apapun bisa mendikte bagaimana seharusnya memetakan aturan hipotesis di dunia ke aturan dalam model. Dalam kata-kata Rosen, sementara

dunia dan model masing-masing internal 'mensyaratkan', tidak ada yang memerlukan dunia dengan model.

Di antara alasan untuk paradoks ini adalah kenyataan bahwa sebagian dari dunia ditangkap oleh model adalah sewenang-wenang dari sebaliknya terbuka. Hal ini terjadi ketika dunia merupakan bagian dari alam sistem, perhatian utama penyelidikan Rosen. Namun pengalaman menunjukkan bahwa bahkan ketika dunia memang sistem yang jelas dan tertutup. Misalnya artefak, perangkat buatan atau sepotong mesin, berbeda.



Gambar 2. 4 Diagram Rosen

Pemodel dapat menghasilkan deskripsi nonequivalent berbeda itu, yaitu, model yang output yang kompatibel dengan set yang sama pengamatan tapi yang strukturnya tidak dipertemukan dengan satu sama lain.

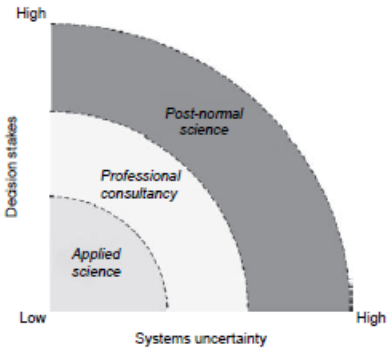
Meskipun hal ini mungkin mengganggu keindahan dan jelas diri bukti hukum-hukum fisika, praktisi pemodelan memiliki datang untuk hidup dengan realitas yang agak tidak menyenangkan yang lebih dari satu model mungkin kompatibel dengan set data yang sama atau bukti. Beberapa telah pergi untuk sebuah kata untuk paradoks ini:

equifinality - Beven (1993, 2001), lihat juga Saltelli et al. (2004, hlm 173-178.) - Yang berarti bahwa berbeda model dapat mengarah pada tujuan yang sama. Lain mengacu pada fenomena sebagai model ketidakpastian.

2.4. 1 Model dan ketidakpastian

Apa yang membuat pemodelan dan penyelidikan ilmiah pada umumnya sangat menyakitkan adalah ketidakpastian. Ketidakpastian bukan kecelakaan dari metode ilmiah. Pemodel dan filsuf ilmu pengetahuan telah memperdebatkan masalah model ketidakpastian panjang lebar (Oreskes et al., 1994). Kebanyakan pemodel hari ini akan mungkin setuju bahwa model tidak dapat divalidasi, dalam arti 'dibuktikan benar'. Sebaliknya, itu adalah lebih dipertahankan dan benar untuk mengatakan bahwa model memiliki secara ekstensif dikuatkan, yang berarti dengan ini bahwa model telah bertahan serangkaian tes - akan mereka formal, konsistensi internal, atau relative.

Kapasitas model untuk menjelaskan atau memprediksi 'dunia' dalam meyakinkan dan cara pelit. Ketika model gagal publik, kontroversi berikutnya dapat menghancurkan untuk pihak ilmiah. Model yang sering digunakan dalam sangat terpolarisasi konteks dan ketidakpastian dapat digunakan instrumental. 'Semua pihak berurusan dengan informasi lingkungan dengan cara selektif, atau bahkan memanipulasinya ', mengamati seorang ilmuwan lingkungan Belanda (Dalam 'Veld t, 2000). ketidakpastian merupakan keprihatinan umum dalam kaitannya dengan sengketa penting atas kesehatan atau lingkungan (Michaels, 2005).



Gambar 2. 5 Diagram ketidakpastian

Singkatnya, model merupakan bagian dari metode ilmiah dan karenanya tunduk perdebatan epistemologis. Sebuah cara membingkai debat hadir-hari di metode ilmiah ditawarkan oleh Post-normal Science (PNS). Dalam PNS satu membedakan antara tiga jenis produksi ilmiah mode, tergantung pada ketidakpastian sistem dan taruhannya terlibat. Dengan penerapan ini untuk pemodelan, persyaratan dan praktek yang berbeda berhubungan:

- Dalam ilmu terapan, ketika model ditulis dan digunakan dalam ditutup konsorsium ahli yang adalah satu-satunya pengguna dari model, misalnya kapan ini digunakan untuk memecahkan dibatasi masalah kinetika.
- Dalam 'konsultasi' ketika model lebih mungkin untuk diteliti, misalnya sebagai bagian dari analisis biaya-manfaat untuk pembangunan jalan baru atau jembatan yang akan mempengaruhi masyarakat;
- Ketika komputasi sensitivitas iklim dalam konteks perubahan global.

Dalam kasus terakhir ini kita berada dalam domain PNS, ilmu pengetahuan di mana (dan model nya) dipanggil untuk memberikan bukti dalam keadaan dari taruhannya bertentangan dan keyakinan. Seperti teori-teori ilmiah, model dapat diberikan silsilah yang membantu kami untuk menilai kualitas mereka. Silsilah mempertimbangkan penggunaan masa lalu dari model, status para pendukungnya, tingkat penerimaan oleh rekan-rekan dan sebagainya (Van der Sluijs, 2002; Craye et al., 2005). Dalam silsilah, kualitas model yang lebih erat terkait dengan 'kebugaran untuk tujuan' - yaitu, dengan tujuan tertentu – dari dengan intrinsik model.

2.4. 2 Mengatur ketidakpastian dan sensitivity analisis

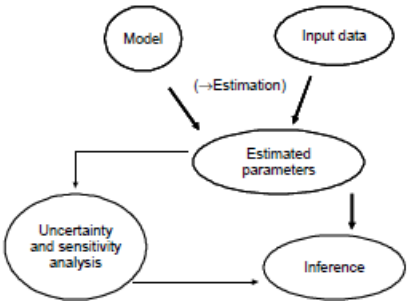
Definisi kita tentang sensitivitas analisis melibatkan model, masukan Model dan output Model. Kita sekarang mencoba untuk mendefinisikan masukan model dalam kaitannya dengan sifat dan tujuan dari model, sebagai serta set-up dari ketidakpastian dan analisis sensitivitas. Sebuah contoh dapat:

- Diagnostik atau prognosis. Dengan kata lain, kita mencoba untuk membedakan antara model yang digunakan untuk memahami hukum dan model yang digunakan untuk memprediksi perilaku sistem yang diberikan undang-undang seharusnya dipahami. model bisa dengan demikian berkisar dari spekulasi liar digunakan untuk bermain (mis model keberadaan intelijen luar bumi) untuk model yang dapat dianggap prediktor akurat dan terpercaya dari sistem (mis control sistem untuk pabrik kimia).

- Data-driven atau hukum-driven. Sebuah model hukum-driven mencoba untuk menempatkan bersama-sama hukum yang diterima yang telah dikaitkan dengan sistem, untuk memprediksi perilakunya. Sebagai contoh, kita menggunakan hukum Darcy dan Ficks 'untuk memahami gerak zat terlarut dalam air yang mengalir melalui berpori medium. Sebuah model data-driven mencoba untuk mengobati zat terlarut sebagai sinyal dan berasal sifat statistik. Para pendukung model data-driven seperti untuk menunjukkan bahwa ini dapat dibangun sehingga menjadi pelit, yaitu untuk menggambarkan realitas dengan minimal parameter disesuaikan (Young et al., 1996).

Model hukum-driven, sebaliknya, lazim overparametrized, sebagai mereka mungkin termasuk hukum lebih relevan daripada jumlah data yang tersedia akan mendukung. Untuk alasan yang sama, model hukum-driven mungkin memiliki kapasitas yang lebih besar untuk menggambarkan sistem dalam keadaan tidak teramati, sementara model data-driven cenderung mematuhi perilaku yang terkait dengan data yang digunakan dalam estimasi mereka. model statistik (seperti hirarkis atau model bertingkat) adalah contoh lain dari model data-driven.

Banyak kategorisasi lain dari model yang mungkin dan definisi masukan Model tergantung pada model khusus yang diteliti. Untuk tujuan ketidakpastian dan analisis sensitivitas kita bebas bisa mengklasifikasikan sebagai masukan segala sesuatu yang dapat mendorong variasi dalam output dari model.



Gambar 2. 6 versi Parametric Bootstrap dari ketidaktentuan dan sensitivity analisis

Pertimbangkan skema pada Gambar 2.6. Di sini kita memiliki pengamatan (diasumsikan bebas dari kesalahan untuk mudahnya) dan model yang parameter diperkirakan dari data. Estimasi dapat mengambil kursus yang berbeda. Biasanya dicapai dengan meminimalkan, misalnya oleh kuadrat, beberapa ukuran jarak antara prediksi model dan data. Pada akhir estimasi langkah, 'terbaik' nilai-nilai parameter serta kesalahan mereka diketahui. pada ini titik kita mungkin mempertimbangkan model 'benar' dan menjalankan analisis ketidakpastian dengan menyebarkan ketidakpastian dalam parameter melalui model, semua cara untuk output Model. Dalam hal ini diperkirakan parameter menjadi faktor.

2. 5 Algoritma *Greedy*

Algoritma untuk permasalahan optimisasi biasanya melalui urutan langkah dengan sekumpulan pilihan pada setiap langkah. Untuk beberapa permasalahan optimisasi menggunakan pemrograman dinamis untuk menentukan pilihan terbaik yang menyelesaikan , sederhana dan algorima yang lebih efisien. Algoritma *greedy*

(Thomas H. Cormen, 2009) selalu membuat pilihan yang melihat peluang yang terbaik pada saatnya. Membuat pilihan optimal secara local dengan berharap bahwa pilihan tersebut dapat membawa pada solusi optimal secara global.

Algoritma greedy tidak selalu menyediakan solusi optimal, tetapi untuk banyak permasalahan yang akan diproses. Melalui permasalahan sederhana tetapi untuk permasalahan nontrivial , aktifitas seleksi permasalahan, dimana algoritma greedy secara efisien melakukan komputasi untuk solusi optimal. Algoritma greedy dengan mempertimbangkan pendekatan pemrograman dinamis dan kemudian dapat membuat pilihan greedy untuk mendatangkan solusi optimal. Element dasar dari pendekatan greedy, memberikan pendekatan langsung untuk membuktikan algoritma greedy benar. Metode greedy merupakan alat yang bertenaga dan bekerja dengan baik untuk permasalahan yang lebih luas.

2.5. 1 Elemen dari strategi *greedy*

Algoritma *greedy* berisikan sebuah solusi optimal dari permasalahan dengan membuat rangkaian dari pilihan. Pada setiap titik keputusan, algoritma membuat pilihan yang terlihat terbaik pada momen. Strategy heuristic tidak selalu memproduksi sebuah solusi optimal, tetapi seperti yang dikatakan pada aktifitas seleksi permasalahan, beberapa telah dilakukan.

Pada seksi ini akan didiskusikan beberapa dari properti metode greedy.

Langkah pada algoritma greedy :

1. menentukan substruktur optimal dari permasalahan

2. membangun solusi rekursi
3. memperlihatkan bahwa jika membuat pilihan greedy, kemudian hanya tertinggal satu subpermasalahan
4. membuktikan bahwa jika selalu aman untuk membuat pilihan greedy
5. membangun algoritma rekursi yang mengimplementasikan strategi greedy
6. mengkonversi algoritma rekursi menjadi algoritma iteratif

Melalui langkah tersebut , kita dapat menyebutkan dengan detail yang baik langkah pemrograman dinamis dari algoritma greedy.

2.5. 2 *Greedy* melawan pemrograman dinamis

Karena keduanya greedy dan strategi pemrograman dinamis mengekploitasi substruktur optimal, kita dapat melakukan membangkitkan solusi pemrograman dinamis untuk permasalahan ketika solusi greedy muncul, bahwa solusi greedy bekerja ketika dalam faktanya solusi pemrograman dinamis diperlukan.

Untuk mengilustrasikan antara kedua teknik tersebut, maka kita melakukan investigasi dari 2 varian dari permasalahan klasik sebagai berikut.

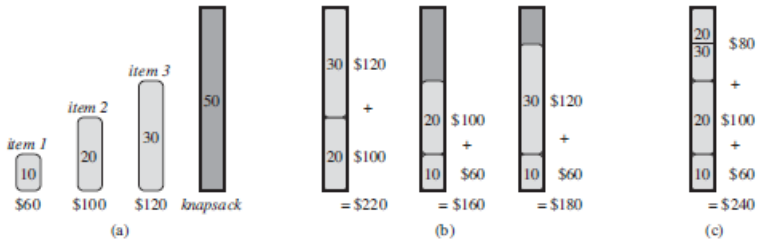
- Permasalahan 0-1 knapsack , Sebuah pencuri toko yang mencari n item. Item I th sangat berharga I dolar dan berat wi pon , dimana I dan wi adalah integer. Pencuri ingin mengambil item yang bernilai dan mengangkutnya sebisa mungkin, tetapi hanya dapat mengangkut w pon pada knapsack untuk integer w. item manakah yang harus dibawa ?

- Permasalahan fractional knapsack , permasalahan nya sama , tetapi pencuri hanya dapat mengambil fraksi dari item, daripada mengambil binary (1-0) pilihan untuk setiap item. Untuk 0-1 knapsack seperti item emas dan untuk fractional knapsack seperti debu emas.

Kedua permasalahan knapsack memperkenalkan property substruktur optimal. Untuk permasalahan 0-1 , diperhitungkan nilai yang dapat diangkut dengan bobot pada W pon. Jika kita mengeluarkan item j dari angkutan, sisa angkutan harus mengangkut setidaknya W – wj dimana pencuri dapat mengambil n – 1 item original terkecuali j. untuk permasalahan fraksional , disebutkan kita mengeluarkan sebuah beban W dari sebuah item j dari angkutan optimal , maka sisa angkutan yang diangkut dengan bobot W – w dimana pencuri dapat mengambil dari n – 1 item original ditambah wj - w pon dari item j.

Walaupun permasalah tersebut serupa , kita dapat memecahkan permasalahan knapsack fractional menggunakan strategy greedy, tetapi kita tidak dapat memecahkan permasalahan 0-1 dengan strategy tersebut. Untuk memecahkan permasalahan fraksional , pertama kita menghitung nilai per pon i=wi untuk setiap item . mematuhi strategy greedy, pencuri memulai dengan mengambil sebanyak mungkin dari item dengan nilai tertinggi per pon. Jika supli sudah penuh dan pencuri dapat membawa lebih banyak, pencuri mengambil sebanyak mungkin dari item dengan nilai tertinggi per pon dan seterusnya sampai pencuri tersebut mencapai batas W . melalui pengurutan item per nilai pon , algoritma greedy berjalan pada O.n lg n / waktu.

Untuk melihat bahwa strategy greedy tidak bekerja pada permasalahan 0-1 knapsack. Dicontohkan bahwa terdapa 3 item dan knapsack dapat menyimpan 50 pon. Item 1 berat 10 pon dan berharga 60 dolar. Item 2 berat 20 pon dan berharga 100 dolar. Item 3 berat 30 pon dan berharga 120 dolar. Lalu harga per pon dari item 1 adalah 6 dolar per pon, dimana lebih besar dari nilai per pon item 2 (5 dolar per pon) atau item 3 (4 dolar per pon). Melalui strategi greedy dimana solusi optimal mengambil item 2 dan 3 .



Gambar 2. 7 Solusi Greedy

Untuk permasalahan fraksional , strategy greedy dimana mengambil item 1 tidak bekerja pada permasalahan 0-1 karena pencuri tidak dapat memenuhi kapasitas knapsack, dan sisa ruang tidak efektif sesuai nilai per pon dari kemampuan angkutannya, kita harus membandingkan solusi pada sub permasalahan yang menyertakan item dengan solusi pada sub permasalahan yang mengecualikan item sebelum kita menentukan pilihan.

2.6. 1 Permutasi

Sebuah Permutasi (Thomas H. Cormen, Introduction To Algorithm . Third Edition, 2009) dari himpunan berhingga S merupakan memerintahkan urutan semua elemen dari S, dimana setiap elemen muncul sekali dalam kemunculannya. Sebagai contoh jika $S = \{a, b, c\}$ kemudian S memiliki 6 permutasi :

abc; acb; bac; bca; cab; cba ;

Dimana terdapat $n!$ Permutasi dari sekumpulan n elemen, dimana kita dapat memilih elemen pertama dari runut dalam n jalan, kedua dalam n-1 jalan , dan ketiga dengan n – 2 jalan dan seterusnya.

2. 6 Sensitivity Analysis dengan menggunakan Greedy

Sensitivity analysis digunakan untuk melihat sebuah bagaimana ketidakpastian dalam hasil keluaran dari sebuah model matematika atau sistem dimana dari sumber yang berbeda dari ketidakpastian dalam masukan. Dimana terdapat beberapa tujuan dari sensitivity analysis seperti :

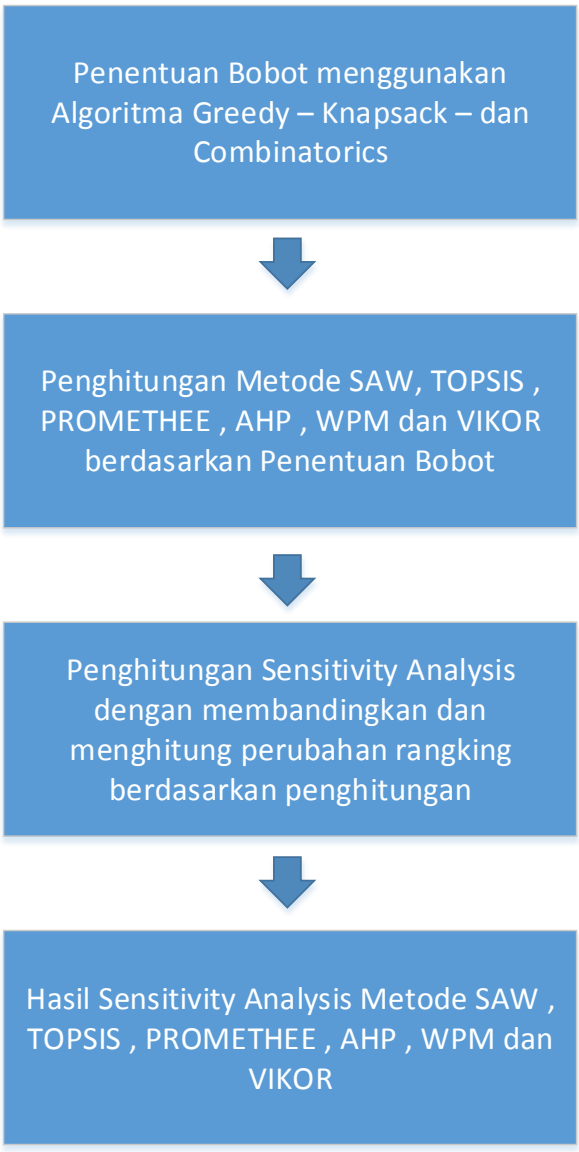
1. Pengujian dari hasil yang baik sebuah model sistem atau ketidakpastian
2. Meningkatkan pemahaman dari keterkaitan antara variabel input dan output sebuah sistem
3. Mencari error di dalam model
4. Simplifikasi model
5. Melakukan kalibrasi model

Melalui sensitivity analysis diperlukan sebuah masukan yang mencangkup seluruh kemungkinan dalam pengaturan parameter sebuah model maka dibutuhkan algoritma greedy di dalam membangkitkan nilai untuk parameter model sehingga diharapkan hasil akan terlihat secara lengkap.

2.7. 1 Menghitung Sensitivity Analysis

Dalam melakukan penghitungan sensitivity analysis diperlukan tahapan di dalam mendapatkan nilai sensitivity analysis dimana terdapat langkah seperti penentuan bobot yang dibangkitkan melalui algoritma greedy dimana bobot harus sama dengan 1 dimana bobot terbagi oleh banyaknya n parameter dengan pecahan yang telah ditentukan kelipatannya seperti 0.1, 0.2 dst hingga 1. Hasil pembangkitan bobot didapatkan dari penggabungan algoritma greedy dan permutasi, dimana algoritma greedy melakukan penukaran pecahan nilai parameter dan algoritma permutasi melakukan permutasi hasil pecahan dari nilai parameter lalu hasil pecahan dari nilai parameter yang telah dipermutasi dijadikan daftar bobot sesuai jumlah parameter dimana daftar bobot tersebut akan dijadikan bobot penghitungan pada metode MCDM. Setelah hasil penghitungan MCDM berdasarkan bobot tersebut dilakukan tahap perbandingan dimana point pembanding adalah penghitungan MCDM dengan bobot dibagi n parameter , BobotPembanding = 1/Nparameter dan hasil penghitungan MCDM tersebut dibandingkan dengan hasil MCDM berdasarkan daftar bobot dari algoritma greedy dan permutasi. Untuk penghitungan Sensitivity Analysis dimana penghitungannya adalah banyaknya perubahan rangking berdasarkan hasil

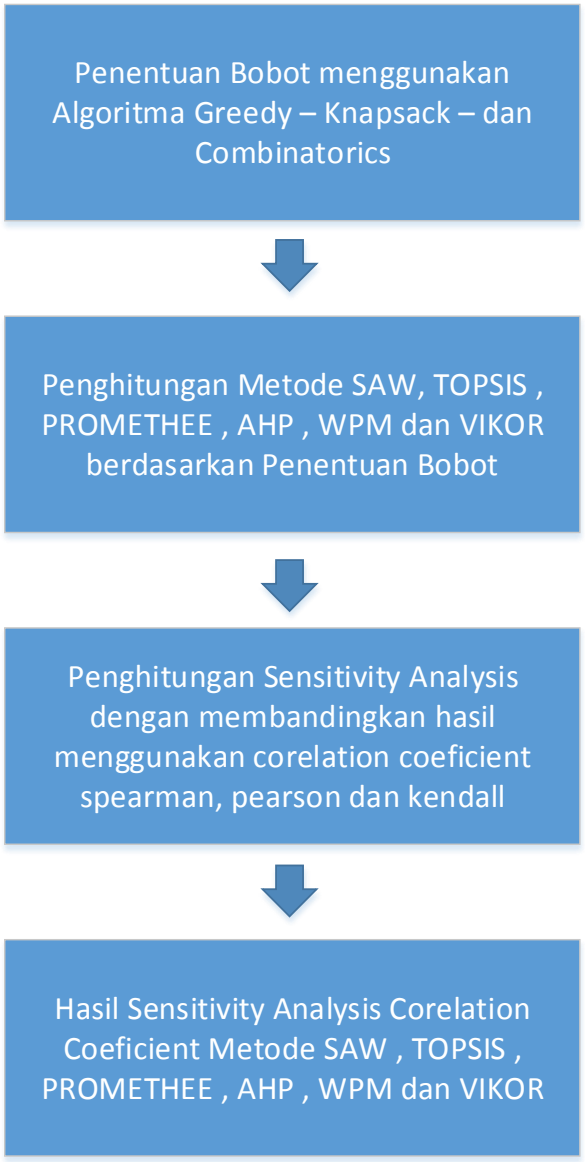
pembanding untuk sebuah metode MCDM dengan hasil MCDM melalui daftar bobot dimana hasil perubahan tersebut dijumlahkan dengan seluruh transaksi/penilaian dan dibagi dengan jumlah transaksi/penilaian atau nilai rata-rata dari hasil perubahan. Langkah penghitungan sensitivity analysis digambarkan sebagai berikut :



Gambar 2. 8 Tahap Penghitungan Sensitivity Analysis Metode MCDM

2.7. 2 Menghitung Sensitivity Analysis dengan *Greedy* dan Corelation Coeficient

Sensitivity Analysis merupakan sebuah analisis terhadap ketidakpastian model dimana sesuai dengan yang dijelaskan sebelumnnya bahwa sensitivity analysis yang dihitung menggunakan bobot yang dibangkitkan oleh algoritma greedy adalah dengan menghitung perubahan rangking, namun tentu hasil akan berbeda apabila dilakukan penghitungan menggunakan correlation coefficient dengan metode spearman , pearson dan kendall dimana proses dilakukan adalah menghitung hasil MCDM pembanding dengan bobotpembanding = 1/nparameter, lalu dilakukan penghitungan MCDM berdasarkan daftar bobot yang dibangkitkan oleh algoritma greedy, setelah itu dilakukan penghitungan correlation coefficient antara hasil MCDM pembanding dan hasil MCDM dengan daftar bobot setelah semua telah dilakukan penghitungan maka dilakukan penjumlahan hasil correlation coeficien dan dibagi dengan jumlah transaksi atau nilai rata-rata dari hasil correlation coefficient. Berikut merupakan tahap penghitungan sensitivity analysis correlation coefficient :



Gambar 2. 9 Tahap Penghitungan Sensitivity Analysis Corelation Coeficient

2. 7 Menganalisa algoritma

Analisa algoritma (Levitin, 2012) diperuntukan memiliki beberapa keunggulan seperti kualitas, yang paling penting adalah efisiensi. Bahkan, ada dua jenis algoritma Efisiensi: efisiensi waktu, menunjukkan seberapa cepat algoritma berjalan, dan efisiensi ruang, menunjukkan berapa banyak memori tambahan yang digunakannya. karakteristik yang diinginkan lain dari algoritma adalah kesederhanaan. Tidak seperti efisiensi, yang dapat didefinisikan secara tepat dan diselidiki dengan matematika, kesederhanaan, seperti kecantikan, adalah untuk gelar yang cukup besar dalam mata yang melihatnya.

Misalnya, kebanyakan orang akan setuju bahwa algoritma Euclid adalah sederhana daripada Prosedur sekolah menengah untuk menghitung gcd (m, n), tetapi tidak jelas apakah algoritma Euclid sederhana dari algoritma bilangan bulat pemeriksaan berturut-turut. Masih, kesederhanaan merupakan ciri algoritma penting untuk berusaha, karena sederhana algoritma lebih mudah untuk memahami dan lebih mudah untuk Program. akibatnya, program yang dihasilkan biasanya mengandung lebih sedikit bug. Ada juga estetika tak terbantahkan dibanding kesederhanaan. Kadang-kadang algoritma sederhana juga lebih efisien dibandingkan alternatif yang lebih rumit. Sayangnya, itu tidak selalu benar, di mana kasus kompromi bijaksana perlu dibuat.

Namun karakteristik yang diinginkan lain dari algoritma adalah umum. Ada, pada kenyataannya, dua masalah di sini: umum dari masalah algoritma memecahkan dan set input diterima. perhatikan bahwa kadang-kadang lebih mudah untuk merancang

sebuah algoritma untuk masalah yang diajukan dalam istilah yang lebih umum. Misalnya, masalah menentukan apakah dua bilangan bulat yang relatif prima, yaitu, apakah pembagi mereka hanya umum adalah sama dengan 1. Hal ini lebih mudah untuk merancang sebuah algoritma untuk masalah yang lebih umum komputasi pembagi umum terbesar dari dua bilangan bulat dan periksa apakah FPB adalah 1 atau tidak. Ada situasi, di mana merancang algoritma yang lebih umum adalah yang tidak perlu atau sulit atau bahkan tidak mungkin. Sebagai contoh, adalah tidak perlu untuk memilah daftar nomor n untuk menemukan median, yang merupakan n nya / 2th elemen terkecil. Memberi contoh lain, rumus standar untuk akar persamaan kuadrat tidak bisa digeneralisasi untuk menangani polinomial derajat sewenang-wenang.

Seperti pada set input, perhatian utama Anda harus merancang sebuah algoritma yang dapat menangani satu set input yang alami untuk masalah yang dihadapi. Sebagai contoh, tidak termasuk bilangan bulat sama dengan 1 input mungkin untuk pembagi umum terbesar algoritma akan cukup wajar. Di sisi lain, meskipun standar rumus untuk akar persamaan kuadrat berlaku untuk koefisien kompleks, kami biasanya tidak menerapkannya pada tingkat umum kecuali kemampuan ini secara eksplisit diperlukan.

Jika Anda tidak puas dengan algoritma efisiensi, kesederhanaan, atau umum, Anda harus kembali ke papan gambar dan mendesain ulang algoritma. Pada kenyataannya, bahkan jika evaluasi Anda adalah positif, itu masih layak mencari solusi algoritmik lainnya.

2.8. 1 Framework Analisis Algoritma

Pada bagian ini, kami menguraikan kerangka umum untuk menganalisis efisiensi algoritma. Terdapat dua jenis efisiensi: efisiensi waktu dan efisiensi ruang. efisiensi waktu, juga disebut waktu kompleksitas, menunjukkan seberapa cepat sebuah algoritma yang bersangkutan berjalan. efisiensi ruang, juga disebut ruang kompleksitas, mengacu pada jumlah unit memori yang diperlukan oleh algoritma selain untuk ruang yang diperlukan untuk input dan output. Pada hari-hari awal elektronik komputasi, baik sumber daya-waktu dan ruang-pada premi. Setengah abad inovasi teknologi tanpa henti telah meningkatkan kecepatan komputer dan ukuran memori beberapa kali lipat.

Sekarang jumlah ruang tambahan yang diperlukan oleh algoritma biasanya bukan dari sebanyak perhatian, dengan peringatan bahwa masih ada, tentu saja, perbedaan antara memori utama cepat, lebih lambat memori sekunder, dan cache. Masalah waktu tidak berkurang cukup dengan tingkat yang sama, namun. Selain itu, pengalaman penelitian telah menunjukkan bahwa untuk sebagian besar masalah, kita dapat mencapai kemajuan yang jauh lebih spektakuler dalam kecepatan daripada di ruang. Oleh karena itu, mengikuti tradisi mapan buku teks algoritma, terutama berkonsentrasi pada efisiensi waktu, tetapi kerangka analisis diperkenalkan di sini adalah berlaku untuk menganalisis efisiensi ruang.

2.8. 2 Mengukur Ukuran Masukan

Mari kita mulai dengan pengamatan jelas bahwa hampir semua algoritma berjalan lebih lama dari input yang lebih besar. Sebagai contoh, diperlukan waktu lebih lama untuk memilah array yang lebih besar, kalikan lebih besar matriks, dan sebagainya. Oleh karena itu, adalah logis untuk menyelidiki efisiensi algoritma ini sebagai fungsi dari beberapa parameter n menunjukkan size.1 input algoritma , dalam kebanyakan kasus, memilih parameter seperti cukup mudah. Sebagai contoh, akan ukuran daftar untuk masalah menyortir, mencari, menemukan daftar terkecil elemen, dan sebagian besar masalah lain yang berhubungan dengan daftar. Untuk masalah mengevaluasi p polinomial (x) = anxn +. . . + A0 derajat n, itu akan menjadi polinomial atau jumlah koefisien, yang lebih besar dengan 1 dari derajat. anda yang melihat dari diskusi yang perbedaan minor tersebut tidak penting untuk efisiensi analisis.

Ada situasi, tentu saja, di mana pilihan parameter yang menunjukkan ukuran masukan tidak peduli. Salah satu contohnya adalah komputasi produk dari dua n × n matriks. Ada dua langkah alami ukuran untuk masalah ini. Pertama dan lebih sering digunakan adalah urutan matriks n. Tapi pesaing lainnya adalah jumlah total elemen N dalam matriks yang dikalikan. (Yang terakhir juga lebih umum karena berlaku untuk matriks yang belum tentu persegi.) Karena ada rumus sederhana yang berkaitan kedua ukuran ini, kita dapat dengan mudah beralih dari satu ke yang lain, tetapi jawaban tentang efisiensi algoritma ini akan secara kualitatif berbeda tergantung pada dua langkah-langkah ini.

Pilihan ukuran metrik yang tepat dapat dipengaruhi oleh operasi algoritma yang bersangkutan. Sebagai contoh, bagaimana kita harus mengukur ukuran masukan ini untuk algoritma pemeriksa ejaan? Jika algoritma meneliti karakter individu input, kita harus mengukur ukuran dengan jumlah karakter; jika bekerja dengan pengolahan kata, kita harus menghitung jumlah mereka di input.

Kita harus membuat catatan khusus tentang pengukuran ukuran input untuk algoritma memecahkan masalah seperti memeriksa primality dari bilangan bulat positif n. Di sini, input hanya satu nomor, dan itu adalah besarnya jumlah ini yang menentukan input ukuran. Dalam situasi seperti itu, adalah lebih baik untuk mengukur ukuran dengan jumlah b bit di representasi biner n ini:

$$b = \log_2 n + 1$$

Metrik ini biasanya memberikan ide yang lebih baik tentang efisiensi algoritma yang bersangkutan.

2.8. 3 Mengukur waktu algoritma

Mengukur waktu algoritma ini berjalan, kita hanya dapat menggunakan beberapa unit standar pengukuran waktu-detik, atau milidetik, dan sebagainya-untuk mengukur waktu berjalan dari program penerapan algoritma. Ada kekurangan yang jelas untuk pendekatan semacam itu, namun: ketergantungan pada kecepatan komputer tertentu, ketergantungan pada kualitas program mengimplementasikan algoritma dan compiler yang digunakan dalam menghasilkan kode mesin, dan sulitnya clocking waktu berjalan yang sebenarnya program.

Karena kita setelah ukuran efisiensi algoritma ini, kami ingin memiliki metrik yang tidak tergantung pada faktor-faktor yang asing. Salah satu pendekatan yang mungkin adalah untuk menghitung berapa kali masing-masing algoritma operasi dijalankan. Pendekatan ini baik berlebihan sulit dan, seperti yang kita akan melihat, biasanya tidak diperlukan. Hal yang harus dilakukan adalah mengidentifikasi yang paling penting operasi algoritma, yang disebut operasi dasar, operasi berkontribusi paling untuk total waktu berjalan, dan menghitung jumlah kali dasar operasi dijalankan.

Sebagai aturan, itu tidak sulit untuk mengidentifikasi operasi dasar dari sebuah algoritma: itu biasanya paling operasi memakan waktu dalam lingkaran terdalam algoritma ini. Misalnya, kebanyakan algoritma pengurutan bekerja dengan membandingkan elemen (kunci) dari daftar yang diurutkan dengan satu sama lain; untuk algoritma tersebut, operasi dasar adalah kunci perbandingan. Sebagai contoh lain, algoritma untuk masalah matematika biasanya melibatkan beberapa atau semua dari empat operasi aritmatika: penambahan, pengurangan, perkalian, dan pembagian. Dari empat, yang paling operasi memakan waktu adalah divisi, diikuti oleh perkalian dan kemudian penambahan dan pengurangan, dengan terakhir.

Dengan demikian, kerangka didirikan untuk analisis efisiensi waktu algoritma ini menunjukkan mengukurnya dengan menghitung jumlah kali algoritma ini operasi dasar dieksekusi pada input berukuran n. Kami akan mencari tahu bagaimana untuk menghitung jumlah tersebut untuk algoritma nonrecursive dan rekursif masing-masing.

Berikut adalah aplikasi penting. Biarkan polisi menjadi waktu eksekusi algoritma ini operasi dasar pada komputer tertentu, dan membiarkan C (n) adalah jumlah kali operasi ini perlu dijalankan untuk algoritma ini.

2.8. 4 Mengukur Pertumbuhan

Mengapa penekanan pada urutan hitungan ini pertumbuhan untuk ukuran masukan yang besar? Sebuah perbedaan dalam menjalankan kali pada input kecil tidak apa benar-benar membedakan efisien algoritma dari yang tidak efisien. Ketika kita harus menghitung, misalnya, pembagi bersama terbesar dari dua angka yang kecil, hal ini tidak segera jelas bagaimana algoritma jauh lebih efisien Euclid dibandingkan dengan dua algoritma lainnya dibahas. bahkan mengapa kita harus peduli yang mereka lebih cepat dan oleh berapa banyak. Hanya ketika kita harus menemukan pembagi bersama terbesar dari dua jumlah besar bahwa perbedaan dalam efisiensi algoritma menjadi baik jelas dan penting. Untuk nilai n yang besar, itu adalah fungsi urutan pertumbuhan yang penting: hanya melihat Tabel 2.2, yang berisi nilai-nilai dari beberapa fungsi penting untuk analisis algoritma.

Besarnya angka dalam Tabel 2.2 memiliki arti mendalam untuk analisis algoritma. Fungsi tumbuh paling lambat di antara ini adalah fungsi logaritma. Ini tumbuh begitu lambat, pada kenyataannya, bahwa kita harus mengharapkan program.

Tabel 2. 2 logaritma pemrosesan algoritma

N	$log_2 n$	n	$n\ log_2 n$	n^2	n^3	2^n	$n!$
10	3.3	10^1	$3.3 . 10^1$	10^2	10^3	10^3	$3.6 . 10^6$
10^2	6.6	10^2	$6.6 .10^2$	10^4	10^6	$3.3 . 10^{30}$	$9.3 . 10^{157}$
10^3	10	10^3	$1.0.10^4$	10^6	10^9		
10^4	13	10^4	$1.3.10^5$	10^8	10^{12}		
10^5	17	10^5	$1.7.10^6$	10^{10}	10^{15}		
10^6	20	10^6	$2.0.10^7$	10^{12}	10^{18}		

Menerapkan algoritma dengan dasar-operasi logaritmik menghitung untuk menjalankan praktis seketika masukan dari semua ukuran realistis. Juga mencatat bahwa meskipun spesifik nilai-nilai count tersebut tergantung, tentu saja, di pangkalan logaritma ini, rumus loga n = loga b logb n memungkinkan untuk beralih dari satu basis ke yang lain, meninggalkan logaritmik count tapi dengan konstanta perkalian baru.

Ini sebabnya kami menghilangkan basis logaritma dan menulis cukup log n dalam situasi di mana kita tertarik hanya dalam rangka fungsi ini pertumbuhan ke dalam konstanta perkalian. Di ujung lain dari spektrum adalah 2n fungsi eksponensial dan faktorial fungsi n! Kedua fungsi ini tumbuh begitu cepat sehingga nilai-nilai mereka menjadi astronomis besar bahkan untuk nilai agak kecil dari n. (Ini adalah alasan mengapa kami tidak termasuk nilai-nilai mereka untuk n> 102 pada Tabel 2.2.)

Sebagai contoh, itu akan mengambil sekitar 4. 1010 tahun untuk komputer membuat triliun (1012) operasi per detik untuk mengeksekusi 2100 operasi. Meskipun ini tak terbandingkan lebih cepat dari itu akan diambil untuk mengeksekusi 100! operasi, masih lebih lama dari 4,5 miliar (4,5. 109) tahun-estimasi umur planet bumi. Ada perbedaan besar antara perintah pertumbuhan fungsi 2n dan n !, namun keduanya sering disebut sebagai "Eksponensial pertumbuhan fungsi" (atau hanya "eksponensial") meskipun fakta bahwa, tegasnya, hanya mantan harus dirujuk ke seperti itu. Garis bawah, yang penting untuk diingat, adalah ini:

“Algoritma yang membutuhkan jumlah eksponensial operasi praktis untuk memecahkan hanya masalah ukuran yang sangat kecil.”

Cara lain untuk menghargai perbedaan kualitatif antara perintah pertumbuhan fungsi pada Tabel 2.2 adalah untuk mempertimbangkan bagaimana mereka bereaksi terhadap, katakanlah, dua kali lipat peningkatan nilai argumen n mereka. Fungsi log2 n kenaikan nilai dengan hanya 1 (karena log2 2n = log2 2 + log2 n = 1 + log2 n); fungsi linear meningkatkan dua kali lipat, fungsi linearithmic n log2 n sedikit meningkat lebih dari dua kali lipat; kuadrat fungsi n2 dan fungsi kubik peningkatan n3 empat kali lipat dan delapan kali lipat, masing-masing (karena (2n) 2 = 4n2 dan (2n) 3 = 8n3); nilai 2n mendapat kuadrat (karena 22N = (2n) 2); dan n! meningkatkan lebih dari itu.

2.8. 5 Efisiensi Kasus Terburuk, Kasus Terbaik dan Kasus Rata-rata

Untuk mengukur Efisiensi algoritma sebagai fungsi dari parameter yang menunjukkan ukuran masukan algoritma. Tapi ada banyak algoritma yang waktu berjalan tergantung tidak hanya pada ukuran masukan tetapi juga pada spesifikasi input tertentu. Mempertimbangkan, sebagai contoh, pencarian sekuensial. Ini adalah algoritma sederhana yang mencari untuk item tertentu (beberapa kunci pencarian K) dalam daftar elemen n dengan memeriksa berturut elemen daftar sampai baik pertandingan dengan kunci pencarian ditemukan atau daftar habis.

Efisiensi kasus terburuk dari algoritma adalah efisiensi untuk terburuk masukan dari ukuran n, yang merupakan masukan (atau input) dari ukuran n yang algoritma menjalankan terpanjang di antara semua kemungkinan input sebesar itu. Cara untuk menentukan efisiensi terburuk dari algoritma adalah, pada prinsipnya, cukup mudah: menganalisis algoritma untuk melihat apa jenis input menghasilkan nilai terbesar dari dasar Operasi ini menghitung C (n) di antara semua kemungkinan input berukuran n dan kemudian menghitung ini nilai terburuk Cworst (n). (Untuk pencarian sekuensial, jawabannya jelas. metode untuk menangani situasi kurang sepele) Jelas, analisis kasus terburuk memberikan informasi yang sangat penting tentang efisiensi algoritma oleh bounding waktu yang berjalan dari atas. di lain kata-kata, ia menjamin bahwa untuk setiap contoh dari ukuran n, waktu berjalan tidak akan melebihi Cworst (n), waktu yang berjalan pada input terburuk.

Efisiensi kasus terbaik algoritma adalah efisiensi yang terbaik-kasus masukan ukuran n, yang merupakan masukan (atau input) dari ukuran n yang algoritma

menjalankan tercepat di antara semua kemungkinan input sebesar itu. Dengan demikian, kita dapat menganalisis kasus terbaik efisiensi sebagai berikut. Pertama, kita menentukan jenis masukan yang menghitung

C (n) akan menjadi yang terkecil di antara semua kemungkinan input berukuran n. (Perhatikan bahwa yang terbaik kasus tidak berarti input terkecil; itu berarti input berukuran n dimana

algoritma menjalankan tercepat.) Kemudian kami memastikan nilai C (n) pada ini yang paling

nyaman. Misalnya, input kasus terbaik untuk pencarian sekuensial adalah daftar

ukuran n dengan elemen pertama mereka sama dengan kunci pencarian; sesuai, Cbest (n) = 1

untuk algoritma ini.

Analisis efisiensi kasus terbaik adalah tidak sepenting yang dari efisiensi terburuk. Tapi itu tidak benar-benar berguna, baik. meskipun kami tidak harus mengharapkan untuk mendapatkan masukan terbaik-kasus, kita mungkin bisa mengambil keuntungan dari fakta bahwa untuk beberapa algoritma kinerja terbaik-kasus yang baik meluas ke beberapa jenis berguna input dekat untuk menjadi yang terbaik-kasus. Misalnya, ada algoritma sorting (insertion sort) yang input kasus terbaik sudah diurutkan array yang algoritma bekerja sangat cepat. Selain itu, efisiensi yang terbaik-kasus memburuk hanya sedikit untuk array hampir-diurutkan. Oleh karena itu, algoritma tersebut mungkin menjadi metode pilihan untuk aplikasi berurusan dengan hampir-diurutkan array.

Dan, tentu saja, jika efisiensi terbaik-kasus algoritma tidak memuaskan, kita bisa langsung membuangnya tanpa analisis lebih lanjut. Itu harus jelas dari diskusi kami, bagaimanapun, bahwa baik terburuk analisis atau kasus terbaik dengan rekan menghasilkan informasi yang diperlukan tentang perilaku algoritma pada "khas" atau "random" input.

Berikut adalah informasi yang efisiensi rata-kasus berusaha untuk diberikan. Untuk menganalisis kasus rata-rata algoritma efisiensi, kita harus membuat beberapa asumsi tentang kemungkinan input berukuran n. Mari kita pertimbangkan pencarian lagi berurutan. Asumsi standar yang yang (A) kemungkinan pencarian yang sukses adalah sama dengan p ($0 \leq p \leq 1$) dan (B) probabilitas pertandingan pertama terjadi di posisi ke-i dari daftar adalah sama untuk setiap i. Di bawah asumsi-validitas yang biasanya sulit untuk memverifikasi, kewajaran mereka meskipun-kita dapat menemukan jumlah rata-rata perbandingan kunci Cavg (n) sebagai berikut. Dalam kasus pencarian berhasil, probabilitas pertandingan pertama terjadi di posisi ke-i dari daftar adalah p / n untuk setiap i, dan jumlah perbandingan yang dibuat oleh algoritma dalam situasi seperti ini. Dalam kasus pencarian gagal, jumlah perbandingan akan n dengan probabilitas pencarian tersebut menjadi (1- p).

2.8. 6 Rekapitulasi dari framework analisis

Waktu dan ruang efisiensi diukur sebagai fungsi dari algoritma ukuran input. efisiensi waktu diukur dengan menghitung jumlah kali algoritma operasi dasar

dijalankan. efisiensi ruang diukur dengan menghitung jumlah unit memori tambahan yang dikonsumsi oleh algoritma.

Efisiensi dari beberapa algoritma mungkin berbeda secara signifikan untuk masukan dari ukuran sama. Untuk algoritma tersebut, kita perlu membedakan antara kasus terburuk, Rata-rata kasus, dan kasus terbaik efisiensi. kepentingan utama kerangka terletak pada urutan pertumbuhan algoritma waktu berjalan (unit memori tambahan yang dikonsumsi) sebagai ukuran input ke tak terhingga.

2.8. 7 Notasi Asymptotic dan kelas dasar efisiensi

Kerangka analisis efisiensi berkonsentrasi pada urutan pertumbuhan hitungan operasi dasar algoritma sebagai Indikator utama dari efisiensi algoritma. Untuk membandingkan dan peringkat perintah tersebut pertumbuhan, ilmuwan komputer menggunakan tiga notasi: O (big oh), Ω (omega besar), dan Θ (Theta besar). Dalam pembahasan berikut, t (n) dan g (n) dapat berupa fungsi non-negatif didefinisikan pada himpunan bilangan. Di konteks kita tertarik, t (n) akan menjadi waktu algoritma ini berjalan (biasanya ditunjukkan dengan yang dasar count operasi C (n)), dan g (n) akan ada beberapa fungsi sederhana untuk membandingkan hitungan dengan.

- **O-notasi** , Sebuah fungsi t (n) dikatakan dalam O (g (n)), dinotasikan t (n) ∈ O (g (n)),

  jika t (n) dibatasi di atas oleh beberapa beberapa konstan g (n) untuk semua n besar, yaitu, jika terdapat beberapa c positif konstan dan beberapa bilangan bulat positif n 0 sehingga t (n) ≤ cg (n) untuk semua n ≥ n0.

- **Ω –notation** , Sebuah fungsi t (n) dikatakan dalam Ω (g (n)), menunjukkan tn) ∈ Ω (g (n)), jika t (n) dibatasi bawah oleh beberapa beberapa konstanta positif dari g (n) untuk semua n besar, yaitu, jika ada ada beberapa c konstanta positif dan beberapa bilangan bulat positif n 0 seperti bahwa t (n) ≥ cg (n) untuk semua n ≥ n0. Berikut adalah contoh dari bukti formal yang n3 ∈ Ω (n2): n3 ≥ n2 untuk semua n ≥ 0, yaitu, kita dapat memilih c = 1 dan n0 = 0.
- **Θ-notasi** , Sebuah fungsi t (n) dikatakan dalam Θ (g (n)), t dinotasikan (n) ∈ Θ (g (n)), jika t (n) dibatasi baik di atas dan di bawah oleh beberapa kelipatan konstanta positif g (n) untuk semua n besar, yaitu, jika ada ada beberapa konstanta positif c1 dan c2 dan beberapa nonnegatif bilangan bulat n 0 sehingga C2G (n) ≤ t (n) ≤ c1g (n) untuk semua n ≥ n0.

2. 8 Mengukur Performa[1]

Sistem komputer telah berubah dari domain eksklusif dari beberapa ilmuwan dan insinyur yang digunakan mereka untuk mempercepat beberapa perhitungan esoteris, seperti menghitung lintasan peluru artileri, misalnya, menjadi begitu umum bahwa mereka tidak diketahui. mereka telah menggantikan banyak dari sistem kontrol mekanik di mobil, sehingga mengurangi biaya sambil meningkatkan efisiensi, kehandalan, dan kinerja. Mereka telah dimungkinkan sebelumnya fiksi ilmiah seperti perangkat seperti telepon seluler. Mereka telah menyediakan waktu berjam-jam

[1] David . J .Lilja. 2004 . Measuring Computer Performance. A Practitioner's guide . Cambridge University Press

hiburan untuk anak-anak mulai usia 1-100. Mereka bahkan telah membawa suara ke kartu ucapan umum. Sebuah konstan sepanjang proliferasi ini dan perubahan, bagaimanapun, telah menjadi kebutuhan bagi pengembang sistem dan pengguna untuk memahami kinerja perangkat berbasis komputer ini.

Sementara mengukur biaya sistem biasanya relatif mudah (kecuali untuk dampak pembaur dari produsen diskon kepada pelanggan khusus), menentukan kinerja sistem komputer seringkali seperti sia-sia. Anehnya, salah satu kesulitan utama dalam mengukur kinerja adalah bahwa orang-orang yang masuk akal sering tidak setuju tentang bagaimana kinerja harus diukur atau ditafsirkan, dan bahkan pada apa yang "kinerja" sebenarnya berarti.

Analisis kinerja yang diterapkan pada ilmu komputer eksperimental dan teknik harus dianggap sebagai kombinasi pengukuran, interpretasi, dan komunikasi dari sistem komputer "kecepatan" atau "ukuran" (kadang-kadang disebut sebagai "kapasitas" nya). Istilah kecepatan dan ukuran yang dikutip dalam konteks ini untuk menekankan bahwa definisi mereka yang sebenarnya sering tergantung pada spesifik situasi. Juga, penting untuk menyadari bahwa kita tidak perlu selalu berurusan dengan sistem yang lengkap. Sering perlu untuk menganalisis sebagian hanya kecil dari sistem independen dari komponen lainnya. Sebagai contoh, kita mungkin tertarik dalam mempelajari kinerja antarmuka jaringan sistem komputer tertentu seperti ukuran memori dari jenis prosesor.

Sayangnya, komponen dari sistem komputer dapat berinteraksi di sangat kompleks, dan sering tak terduga. salah satu tanda seorang analis kinerja komputer ahli

adalah bahwa ia dapat mengukur selain interaksi ini untuk mengetahui pengaruh kinerja karena hanya untuk komponen tertentu.

Salah satu tugas yang paling menarik dari analis kinerja dapat mencari tahu bagaimana untuk mengukur data yang diperlukan. Dosis besar kreativitas mungkin diperlukan untuk mengembangkan teknik pengukuran yang baik yang mengacaukan sistem sesedikit mungkin sambil memberikan hasil akurat direproduksi. Setelah data yang diperlukan telah dikumpulkan, analis harus menginterpretasikan hasilnya dengan menggunakan teknik statistik yang sesuai. Akhirnya, bahkan sangat baik pengukuran ditafsirkan secara statistik sesuai tidak berguna praktis untuk siapa pun kecuali mereka dikomunikasikan dengan cara yang jelas dan konsisten.

2.9. 1 Metrik Performa

Sebelum kita bisa mulai memahami setiap aspek dari kinerja sistem komputer, kita harus menentukan hal-hal apa yang menarik dan berguna untuk mengukur. Karakteristik dasar dari sebuah sistem komputer yang kita biasanya perlu mengukur adalah

- menghitung berapa kali suatu peristiwa terjadi
- durasi beberapa interval waktu
- ukuran beberapa parameter

Misalnya, kita mungkin perlu untuk menghitung berapa kali prosesor memulai permintaan input / output. Kami juga mungkin tertarik dalam berapa lama setiap

permintaan tersebut membutuhkan. Akhirnya, itu mungkin juga berguna untuk menentukan jumlah bit yang ditransmisikan dan disimpan.

Dari jenis nilai yang terukur, kita dapat memperoleh nilai yang sebenarnya yang ingin kita gunakan untuk menggambarkan kinerja sistem. Nilai ini disebut metrik kinerja.

2.9. 2 Metrik Performa System

Berbagai macam metrik kinerja telah diusulkan dan digunakan di bidang komputer. Sayangnya, banyak dari metrik ini tidak baik dalam arti yang didefinisikan di atas, atau mereka sering digunakan dan ditafsirkan salah. Subbagian berikut menggambarkan banyak dari metrik umum dan mengevaluasi mereka terhadap karakteristik di atas metrik kinerja yang baik:

- **Clock rate**, Indikasi yang paling menonjol dari kinerja sering frekuensi clock prosesor pusat.
- **MIPS**, keluaran atau eksekusi kinerja tingkat metrik adalah ukuran dari jumlah perhitungan dilakukan per satuan waktu. MIPS adalah singkatan jutaan instruksi dieksekusi per detik.
- **MFLOPS**, MFLOPS adalah singkatan jutaan operasi floating-titik dieksekusi per detik, mendefinisikan operasi aritmatika pada dua jumlah floating point menjadi unit dasar dari "jarak"
- **SPEC**, SPEC adalah Sistem Evaluasi Kinerja Koperasi. Kelompok ini mengidentifikasi serangkaian integer dan floating point program benchmark

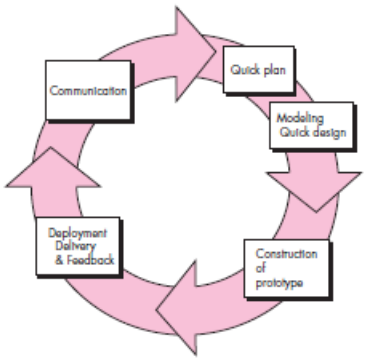
yang dimaksudkan untuk mencerminkan cara yang paling sistem komputer kelas workstation benar-benar digunakan.

- **QUIPS**, QUIPS adalah kualitas solusi sebagai indikasi lebih bermakna dari tujuan akhir pengguna.
- **Waktu Eksekusi**, Seberapa cepat sebuah program yang diberikan dijalankan. Teknik dasar untuk mengukur waktu dalam sebuah sistem komputer analog dengan menggunakan stopwatch untuk mengukur waktu yang dibutuhkan untuk melakukan beberapa acara.

2. 9 Pengembangan Perangkat Lunak dengan Metode *Prototipe* dan *Iterative*

Metode *Prototyping* (Pressman, 2010) merupakan kegiatan pengembangan perangkat lunak ketika pelanggan mendefinisikan sekumpulan dari objektif umum untuk perangkat lunak, tetapi tidak mengidentifikasi kebutuhan yang mendetail untuk fungsi dan fitur. Pada kasus lain pengembang tidak begitu yakin untuk efisiensi dari algoritma, adaptasi sistem operasi atau bentuk dari interaksi computer dan manusia yang harus dibuat.

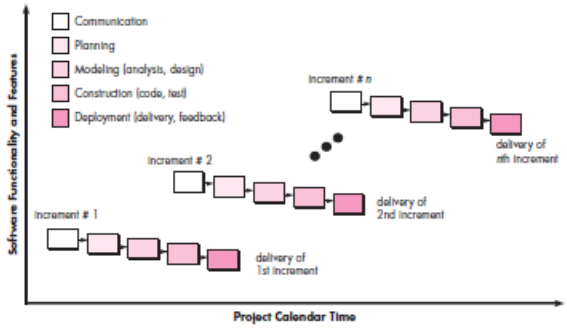
Melalui *prototyping* dapat digunakan sebagai proses berdiri sendiri, biasanya digunakan sebagai teknik yang dapat diimplementasikan dengan kontek setiap model proses. Melalui sikap yang diaplikasikan, paradigm *prototyping* memberikan pengembang dan pemeran bisnis / pengguna menjadi lebih mengerti apa yang harus dibuat ketika kebutuhan masih kurang lengkap/tidak jelas.



Gambar 2. 10 Proses Pengembangan Perangkat Lunak Prototipe

Model *incremental/Iterative* (Pressman, 2010) mengkombinasikan elemen dari aliran proses linear dan parallel . model incremental mengaplikasikan langkah urutan secara linear dalam kalender progress waktu. Setiap langkah linear produk memberikan "increment/penambahan" dari perangkat lunak yang serupa untuk produk increment melalui proses aliran *evolutionary*.

Dimana langkah yang dilakukan bertahap melalui iterasi komunikasi/kebutuhan, perencanaan , pemodelan dan analysis, pembangunan dan pembangunan dimana akan menghasilkan produk versi dan dilakukan iterasi berikutnya untuk menghasilkan versi produk dengan penambahan fitur/modul.

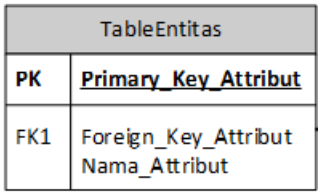
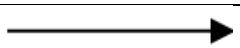


Gambar 2. 11 Proses Pengembangan Perangkat Lunak Iterative

2. 10 *Conceptual* Diagram

Untuk melakukan perancangan database yang akan dibangun dilakukan pemodelan dengan menggunakan diagram *conceptual*. Diagram *Conceptual* (Conceptual Diagram, n.d.) adalah suatu teknik pemodelan seccara grafis yang menjelaskan attribute dari entitas dan relasi dari entitas attribut. Standar IDEF1x dikembangkan untuk pengerjaan desain basis data relasional. Standar IDEF1x digunakan untuk konstruksi dari skema konseptual yang merepresentasikan struktur dari data dalam kontek dari sistem. IDEF1x standar dibangun secara khusus untuk konstruksi dari struktur basis data dan untuk pekerjaan memasukan definisi data. Pada entitas IDEF1x merupakan objek secara keseluruhan dari objek pada dunia nyata untuk property kunci. Setiap objek konkrit pada kasus untuk realisasi dari entitas. Berikut merupakan penjelasan simbol *conceptual* diagram :

Tabel 2. 3 Simbol-simbol pada *Conceptual* Diagram.

No	Simbol	Keterangan
1.		Tabel Entitas Merupakan Entitas untuk menggambarkan Objek Data. Attribut merupakan anggota dari Entitas untuk menyimpan informasi data. Primary Key (PK) untuk menggambarkan entitas attribute kunci. Foreign Key (FK) untuk menggambarkan entitas attribute Pasangan Attribut Kunci pada entitas Lain.
2		Relasi dimana menggambarkan keterkaitan antara Primary Key dengan Foreign Key.

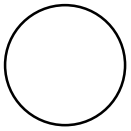
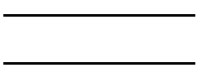
2. 11 Data Flow Diagram

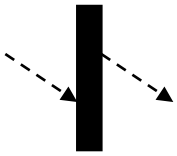
Untuk menganalisis sistem yang dibuat dilakukan pemodelan dengan menggunakan *Data Flow Diagram* (DFD) (Bentley, 2006). DFD adalah suatu teknik pemodelan secara grafis yang menggambarkan aliran data dalam sistem serta fungsi-fungsi (proses) yang terlibat dalam transformasi aliran data tersebut.

Selain dapat memberikan informasi tambahan yang digunakan selama tahap analisis, DFD juga digunakan untuk merepresentasikan sistem atau perangkat lunak pada berbagai tingkatan abstraksi. DFD dapat dibagi menjadi beberapa level yang menggambarkan penambahan aliran informasi dan fungsionalitas yang lebih rinci.

DFD level 0 (diagram konteks) merepresentasikan hubungan perangkat lunak/sistem dengan lingkungan yang berkaitan dengan sistem tersebut sebagai satu proses dengan data masukan dan keluaran digambarkan sebagai panah yang masuk dan keluar proses. Selanjutnya pada level yang lebih tinggi (1,2,..dst), proses tersebut dipecah-pecah untuk memperoleh aliran data dan proses yang lebih rinci. Tabel 2.3 menjelaskan simbol dan notasi yang digunakan di dalam perancangan DFD.

Tabel 2. 4 Simbol-simbol pada Diagram Alir Data (*Data Flow Diagram*).

No	**Simbol**	**Keterangan**
1.		*External entity* (kata benda), sebuah elemen sistem (misalnya perangkat keras, seseorang, program yang lain) atau sistem yang lain yang menghasilkan informasi bagi transformasi oleh perangkat lunak atau menerima informasi yang dihasilkan oleh PL.
2.		Proses (kata kerja), menggambarkan kegiatan yang dilakukan oleh sistem, dapat berupa prosedur yang memanipulasi atau mengolah data.
3.		*Data Store* (kata benda), merupakan tempat penyimpanan data untuk direferensi atau diolah lagi lebih lanjut.

No	Simbol	Keterangan
4.		*Data Flow* (kata benda), menggambarkan aliran data yang menunjukkan transportasi data atau informasi.
5.		*Control Flow* (kata benda), menggambarkan aliran kontrol data atau kejadian, dapat berupa nilai Boolean
6.		*Control Bar*, merupakan acuan untuk CSPEC (*control specification*) yang menjelaskan kebiasaan sistem dan mendefinisikan bagaimana proses diaktifkan sebagai konsekuensi dari suatu kejadian.

2. 12 Kamus Data

Kamus data (Bentley, 2006) adalah kumpulan elemen-elemen atau simbol-simbol yang digunakan untuk membantu dalam penggambaran atau pengidentifikasian setiap *field* atau file di dalam sistem. Dapat dikatakan juga kamus data atau *data dictionary* atau disebut juga dengan istilah *system data dictionary* adalah katalog fakta tentang data dan kebutuhan-kebutuhan informasi dari suatu sistem informasi.

Dengan menggunakan kamus data, analis sistem dapat mendefinisikan data yang mengalir di sistem dengan lengkap.Kamus data dibuat pada tahap analisis sistem

dan digunakan baik pada tahap analisis maupun pada tahap perancangan sistem.Pada tahap analisis, kamus data dapat digunakan sebagai alat komunikasi antara analis sistem dengan pemakai sistem tentang data yang mengalir di sistem, yaitu tentang data yang masuk ke sistem dan tentang informasi yang dibutuhkan oleh pemakai sistem.

Pada tahap perancangan sistem, kamus data digunakan untuk merancang *input*, merancang laporan-laporan, dan *database*. Kamus data dibuat berdasarkan arus data yang ada di DFD *(Data Flow Diagram).* Arus data pada DFD memiliki sifat global, hanya ditunjukkan oleh nama arus data. Untuk keterangan lebih lanjut tentang struktur dari suatu arus data di DFD secara lebih terinci dapat dilihat di kamus data pada tabel 2.2 pada halaman 2-14 berikut.

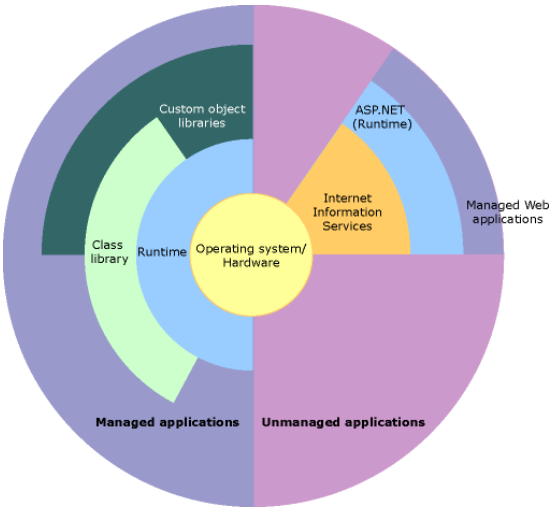
Tabel 2. 5 Simbol-simbol kamus data.

Notasi	**Keterangan**
=	Terdiri dari / didefinisikan sebagai
+	Dan
(…)	Opsional
{ …}	Iterasi/ pengulangan
[…]	Pemillihan dari beberapa alternative
…	Komentar
@	Identifikasi atribut kunci
I	Pemisahan pada pemilihan (atau)

2. 13 .NET Framework

.NET Framework™ 4.0 (Microsoft Developer Network,.NET Framework, n.d.) merupakan bagian komponen integral dari Window yang mendukung pembangunan dan menjalankan generasi aplikasi dan *web service* berikunya. .NET Framework™ 4.0 di rancang untuk memenuhi :

1. Untuk mendukung kebutuhan pemrograman berorientasi objek dimana kode objek disimpan dan dieksekusi secara lokal, namun terdistribusi bersama internet atau dapat dieksekusi dengan dipanggil.
2. Untuk mendukung kebutuhan eksekusi kode yang meminimasi pengimplementasian perangkat lunak dan konflik penamaan versi.
3. Untuk mendukung kebutuhan eksekusi kode yang memberikan eksekusi kode secara aman termasuk kode yang dibuat oleh pihak ketiga.
4. Untuk mendukung kebutuhan eksekusi kode yang mengeliminasi permasalahan performa dari skrip atau kebutuhan yang diintepretasikan.
5. Untuk membuat pengalaman pengembang konsisten dari banyak tipe aplikasi seperti aplikasi berbasis *Window* dan Web.
6. Untuk membangun komunikasi dalam standar industri di dalam meyakinkan bahwa kode berbasis pada .NET Framework™ 4.0 yang dapat diitegrasikan dengan kode lain.



Gambar 2. 12 Kontek .NET Framework

.NET Framework™ 4.0 dapat ditanam dengan komponen yang tidak terkelola dan dijalankan oleh CLR ke dalam prosesnya dan inisiasi untuk dieksekusi dari kode yang terkelola, sehingga membuat kebutuhan perangkat lunak dapat mengeksploitasi fitur antara kode yang terkelola dan tidak terkelola..NET Framework™ 4.0 tidak hanya dapat memberikan beberapa eksekusi secara lokal namun juga mendukung pembangunan eksekusi pihak ketiga. .NET Framework™ 4.0 memiliki 2 komponen utama :

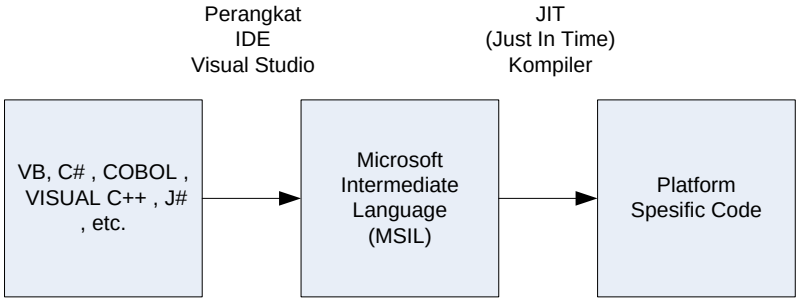
1. ***Common Language Runtime*(CLR)**, CLR merupakan fondasi dari .NET Framework. CLR adalah agen yang mengatur kode pada saat eksekusi, memberikan pelayanan utama seperti manajemen memori, manajemen *thread*, dan *remoting*. Dimana memberikan tipe yang aman dan bentuk lain dari keakuratan kode sehingga memberikan keamanan dan ketangguhan. Manajemen kode merupakan prinsip dasar dari CLR, kode yang ditargetkan

untuk dijalankan merupakan kode yang terkelola, kode yang tidak ditargetkan untuk dijalankan disebut kode yang tidak terkelola.

CLR mengelola memori, eksekusi *thread*, eksekusi kode, verifikasi keamanan kode, kompilasi, dan pelayanan sistem lainnya. CLR memiliki fitur yang menentukan untuk mengelola kode yang akan dijalankan pada CLR. Dengan kebutuhan akan keamanan, pengelolaan komponen merupakan bentuk berbagai kepercayaan, bergantung kepada banyaknya faktor yang disertakan pada internet, jaringan *enterprise* atau komputer lokal. CLR membantu keamanan akses kode, pengguna dapat percaya terhadap eksekusi program yang tertanam di dalam halaman web yang menjalankan animasi pada layar atau suara lagu tetapi hanya dapat mengakses data pribadi, sistem file atau jaringan.Fitur keamanan dari CLR dapat memungkinkan legitimasi pengimplementasian perangkat internet untuk pengecualian kekayaan fitur.

CLR membantu ketangguhan kode dengan mengimplementasikan tipe kode dan verifikasi kode infrastruktur yang disebut CTS (*common type system*).CTS memastikan semua kode yang terkelola mendeskripsikan dirinya.Pada bahasa kompiler Microsoft® dan pihak ketiga membangkitkan kode yang terkelola sehingga dapat diterima oleh CTS.Kode yang terkelola dapat dikonsumsi oleh tipe dan instan lainnya, memberikan kejelasan tipe data dan *type safety*. Kebutuhan yang terkelola dari CLR, mengeliminasi berbagai isu perangkat lunak.Dengan manajemen memori otomatis mengatasi permasalahan *error* pada aplikasi, keterlambatan memori dan kesalahan referensi memori.

CLR membantu percepatan produktifitas pengembang.Programer dapat menulis aplikasi dengan pilihan bahasa untuk digunakan di dalam pembangunan.memiliki keuntungan CLR, pustaka kelas dan komponen yang ditulis di dalam bahasa lain oleh pengembang lain. Kompiler perusahaan dapat memilih target eksekusi. Kompiler bahasa target .NET *Framework* membuat fitur dari .NET Framework™ 4.0 agar tersedia dari kode yang telah ditulis dengan bahasanya, sehingga mempermudah proses migrasi aplikasi. Pada gambar 2.6 berikut menjelaskan proses eksekusi pada .NET Framework (Platt, 2003).



Gambar 2. 13 Proses eksekusi pada .NET *Framework*

CLR dirancang untuk perangkat lunak masa depan dan mendukung perangkat lunak sekarang dan masa yang akan datang. Integrasi antara kode terkelola dan tidak terkelola, sehingga memungkinkan pengembang dapat menggunakan komponen COM dan DLL.

CLR di rancang untuk meningkatkan performa.Sejak CLR memberikan banyak standar pelayanan eksekusi, kode yang terkelola tidak pernak diintepretasikan. Fitur kompilasi JIT (*Just In Time*) memungkinkan semua kode yang terkelola

dijalankan ke dalam *native machine language*, sistem dimana kode tersebut dieksekusi. Manajer memori menyingkirkan kemungkinan memori yang terfragmentasi dan meningkatkan memori lokal untuk meningkatkan performa. CLR dapat ditanamkan pada *server-side* berperforma tinggi, seperti Microsoft® SQL Server™ dan *Internet Information Services* (IIS). Infrastruktur ini memungkinkan untuk menggunakan kode yang terkelola dalam menulis *bussines logic*, selama masih menggunakan performa dari *enterprise server industry* terbaik yang mendukung eksekusi aplikasi melalui *hosting*.

2. **.NET *Framework class library*,** merupakan komponen utama .NET Framework™ 4.0, koleksi berorientasi objek dari tipe yang bisa digunakan kembali sehingga dapat digunakan untuk membangun aplikasi secara tradisional *command line* atau GUI (*Graphical User Interface*) yang merupakan basis dari inovasi terbaru seperti web *form,* XMLdan web *service*.

   .NET *Framework* pustaka kelas merupakan koleksi dari tipe yang dapat digunakan kembali dan terintegrasi dengan CLR.Pustaka kelas berorientasi objek, memberikan tipe dari kode yang terkelola dan dapat menurunkan fungsionalitas.Dengan fitur baru pada .NET Framework™ 4.0 yang mereduksi waktu asosiasi sehingga komponen pihak ketiga dapat berintegrasi dengan kelas pada .NET Framework™ 4.0.

   Koleksi kelas .NET Framework™ 4.0 mengimplementasikan sekumpulan antarmuka yang dapat digunakan untuk membangun koleksi kelas

pribadi.Koleksi kelas dapat dicampur dengan kelas yang terdapat pada .NET Framework™ 4.0.

Pustaka kelas berorientasi objek pada tipe .NET Framework™ 4.0 memungkinkan untuk melakukan beberapa pekerjaan program, termasuk manajemen *string*, koleksi data, konektifitas basis data, dan akses file. pustaka kelas termasuk tipe yang mendukung varietas dari spesialisasi skenario pembangunan. .NET Framework™ 4.0 dapat membantu pengembangan tipe aplikasi dan pelayanan :

a. Aplikasi *Console*.
b. Aplikasi *Windows* GUI (Windows *Forms*).
c. Aplikasi *Windows Presentation Foundation* (WPF).
d. Aplikasi ASP.NET.
e. Web *services*.
f. *Windows services*.
g. Aplikasi SOA (*Service Oriented Architecture*) dengan *Windows Communication Foundation* (WCF).
h. Aplikasi *Workflow* dengan *Windows Workflow Foundation* (WWF).

2. 14 Perangkat Lunak Aplikasi (Software)

Software aplikasi merupakan suatu perangkat yang memungkinkan pemakai(*user*) memahami sistem komputer. Berikut beberapa contoh dari *software* aplikasi :

1. **Visual Studio .NET™** merupakansalah satu peralatan pemprograman dengan menggunakan bahasa komputer tingkat tinggi. Bahasa Pemprograman adalah perintah-perintah atau instruksi untuk melakukan tugas-tugas tertentu yang dimengerti oleh komputer. Untuk melakukan tugas-tugas tertentu bahasa pemprograman VS .NET dikembangkan oleh Microsoft, Merupakan salah satu bahasa pemprograman yang *Object Oriented Program* (OOP) atau pemprograman yang berorientasi pada objek. Kata "Visual" menunjukkan cara yang digunakan untuk membuat *Graphical User Interface* (GUI). Tidak perlu menuliskan instruksi pemprograman dalam kode-kode baris hanya untuk membuat sebuah rancangan *Form*/Aplikasi. Tetapi dengan sangat mudah yakni kita cukup melakukan *Drag and drop object-object* yang akan kita gunakan. Merupakan bahasa pemrograman yang biasa digunakan pada lingkungan aplikasi bisnis.
2. **Microsoft SQL Server 2012**

   Microsoft SQL Server 2012™ merupakan basis data sebagai media untuk menyimpan dan administrasi data berkonsolidasi dengan server dan memuat objek bisnis data dari aplikasi melalui aplikasi pengelolaan basis data Sql Server 2012™ Managemen Studio. Perlengkapan studio tersebut dirancang untuk membantu pengembang aplikasi bisnis dalam berkerja dengan basis data Microsoft SQL Server™.