OPTIMASI PEMILIHAN FITUR MENGGUNAKAN ALGORITMA *BRANCH AND BOUND*

TUGAS AKHIR

Disusun Oleh :

SUCI HATINING RINI
5100 100 002

JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2005

OPTIMASI PEMILIHAN FITUR MENGGUNAKAN ALGORITMA *BRANCH AND BOUND*

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan

Memperoleh Gelar Sarjana Komputer

Pada

Jurusan Teknik Informatika

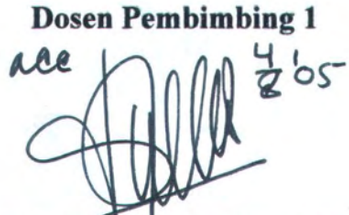
Fakultas Teknologi Informasi

Institut Teknologi Sepuluh Nopember

Surabaya

Mengetahui/Menyetujui,

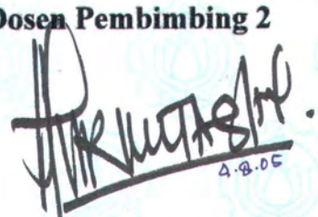
Dosen Pembimbing 1



Rully Soelaiman, S.Kom, M.Kom

NIP. 132 085 802

Dosen Pembimbing 2



Diana Purwitasari, S.Kom

NIP. 132 306 545

Surabaya

Juli 2005

ABSTRAK

Suatu fitur sangat diperlukan untuk mengindentifikasikan suatu objek. Fitur-fitur optimal yang bisa diketahui dari suatu objek akan mempermudah dan mempercepat proses indentifikasi objek tersebut. Fitur yang sedikit akan mempermudah dalam menentukan daerah keputusan (decision regions). Untuk itu perlu dilakukan pemilihan fitur yang paling optimal dari fitur-fitur yang ada.

Metode Branch and Bound merupakan salah satu metode yang digunakan untuk pemilihan fitur optimal. Algoritma ini telah mengalami beberapa perbaikan sehingga ada empat algoritma yang telah dihasilkan yaitu algoritma Branch and Bound dasar (BBB), Improved Branch and Bound (IBB), Branch and Bound Partial Prediction (BBPP) dan Fast Branch and Bound (FBB). Algoritma BBPP dan FBB merupakan algoritma yang paling efisien dalam melakukan penghitungan kriteria. Kedua algoritma ini menggunakan suatu mekanisme prediksi untuk mengurangi jumlah penghitungan evaluasi kriteria.

Uji coba dilakukan menggunakan data sintetik dengan variasi fungsi kriteria yang digunakan telah didefinisikan. Uji coba dilakukan sebanyak lima kali dengan lima fungsi kriteria yang berbeda dan hasil dari uji coba ini menunjukkan bahwa algoritma Branch and Bound sangat dipengaruhi oleh fungsi kriteria dan data yang digunakan, prediksi yang digunakan pada algoritma BBPP dan algoritma FBB menyebabkan performance kedua algoritma tersebut lebih baik daripada algoritma IBB maupun BBB, kondisi terburuk terjadi bila fungsi kriteria yang digunakan menyebabkan semua fitur menghasilkan nilai kriteria yang sama sehingga tidak ada cabang yang mengalami cut off dan nilai kriteria tergantung pada fungsi kriteria yang digunakan, bukan pada ukuran subset di tiap level tree.

KATA PENGANTAR

Alhamdulillahirabbil'aalamiin, segala puji syukur hanya kepada Allah yang telah mencurahkan karunia-Nya kepada penulis sehingga penulis dapat menyelesaikan Tugas Akhir berjudul :

"Optimasi Pemilihan Fitur menggunakan Algoritma *Branch and Bound*"

Tugas Akhir ini merupakan salah satu syarat untuk menyelesaikan program strata satu (S-1) pada jurusan Teknik Informatika, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember Surabaya.

Penulis menyadari bahwa Tugas Akhir ini tidak lepas dari bantuan banyak pihak, maka dalam kesempatan ini, penulis ingin menyampaikan penghormatan dan terimakasih kepada pihak-pihak yang telah memberi bantuan baik itu berupa moril maupun material secara langsung maupun tidak langsung kepada :

1. Bapak Prof. Dr. Ir. Arif Djunaidy, M.Sc, selaku Ketua Dekan Fakultas Teknologi Informasi, ITS.
2. Bapak Yudhi Purwananto, S.Kom, M.Kom, selaku Ketua Jurusan Teknik Informatika, FTIF, ITS.
3. Bapak Rully Soelaiman, S. Kom, M. Kom, selaku Dosen Pembimbing, yang selalu sabar memberikan bimbingan, nasehat, petunjuk, semangat, motivasi, dan atas segala bantuan yang diberikan kepada penulis hingga tugas akhir ini selesai.
4. Ibu Diana Purwitasari, S. Kom, selaku Dosen Pembimbing, yang selalu memberikan bimbingan, nasehat, masukan, saran, semangat dan atas

semua bantuan yang diberikan kepada penulis hingga tugas akhir ini selesai.

5. Bapak Prof. Dr. Ir. Supeno Djanali, Wahyu Suadi, S. Kom, M. Kom, Ir. Hakim Godzali, Ir. M. Husni selaku Dosen Wali penulis yang selalu membantu penulis.
6. Bapak dan Ibu dosen Jurusan Teknik Informatika – ITS yang telah dengan sabar memberikan ilmunya selama Penulis menempuh kuliah.
7. Seluruh staf dan karyawan Jurusan Teknik Informatika – ITS, Pak Yudi, Pak Narno, Mbak Eva, dan karyawan-karyawan lain.
8. Mase yang selalu membantu penulis sewaktu-waktu, selalu menghibur penulis disaat sedih, selalu memberi semangat kepada penulis agar segera lulus. Terimakasih atas kompinya ☺.
9. Papa dan Mama tersayang yang sudah mendidik dan mendoakan penulis dengan penuh kesabaran. Semoga Papa dan Mama selalu mendapatkan kemuliaan, kesabaran, kebahagiaan didunia dan akhirat, serta selalu dalam lindungan Allah SWT, Amien.
10. Kakak-kakakku yang selalu memberi semangat kepada penulis agar segera lulus.
11. Mbak Wiwik yang selalu mendoakan, mendampingi dan membantu penulis disaat-saat penting.
12. Teman-teman satu kosan, Titin dan Wida, yang selalu memberi semangat kepada penulis untuk mengerjakan tugas akhir ini. SEMANGAT...!!! ☺

13. Ibunya Titin, kakaknya Titin, Mas Wawan dan Mbak Pipit, yang telah memberikan bantuan kepada penulis, terutama pada hari H sidang TA. Terimakasih atas jalan-jalannya setelah sidang TA ☺.
14. Erma, Arida, terimakasih atas semua doa dan bantuannya sehingga penulis bisa lulus. AYO SEMANGAT !!!!! Cepetan lulus ya..... CHAYO ERMA.....CHAYO A..... ☺.
15. Yunita, terimakasih atas semua bantuannya yang telah diberikan kepada penulis, terutama tutorialnya GUI Matlab ☺.
16. Teman-teman seperjuangan, Dian, Alifah, Nindi, Fetty, Ruli, Edy'01, Hatfi'01, Warna Agung'99 yang telah membantu dan memberi semangat kepada penulis.
17. Delis, Chery'99, Anggit dan Rontog'99, thank's kompinya ya..... ☺.
18. Mas Anank'96, terimakasih atas bantuannya *spell check* buku TA penulis.
19. Semua penghuni Lab IBS, Fauzan, Galih, Mahmud, Angga, dan teman-teman lain, terimakasih atas bantuannya selama penulis sibuk mengerjakan tugas akhir ini di Lab.
20. Mas Ajun, yang selalu memberi masukan, saran, semangat kepada penulis.
21. Teman-teman C10, Tyarso, Roy, Roni Ext, Reni, Hamah, Dhanti, Okti, Shofi, dan yang lainnya, terimakasih atas dukungan, dan bantuannya selama penulis berkuliah di teknik informatika ini, terimakasih atas pertemanan yang sudah terjalin, semoga pertemanan ini tidak pernah putus.

22. Yeni GK 44, terimakasih telah memberi semangat, dukungan dan sedikit sarapan kepada penulis disaat detik-detik menjelang sidang TA ☺.
23. Mbok, Mbak Is, Mbak Nur, yang telah membantu penulis, terutama menjaga kamar penulis agar listriknya tidak dimatikan.
24. Serta rekan-rekan yang tidak dapat disebutkan satu persatu, yang telah membantu penulis dari awal kuliah hingga tersusunnya tugas akhir ini. Terimakasih.....

Penulis berharap semoga apa yang telah penulis uraikan dalam tugas akhir ini dapat bermanfaat dan dapat dikembangkan lebih lanjut untuk perkembangan ilmu pengetahuan dan teknologi dimasa mendatang.

Surabaya, Juli 2005

Suci Hatining Rini

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

Pada bab ini dijelaskan mengenai latar belakang yang mendasari pembuatan tugas akhir ini, permasalahan, batasan masalah, tujuan dari pembuatan perangkat lunak, metodologi pembuatan serta sistematika penulisan buku tugas akhir.

1.1 LATAR BELAKANG

Suatu objek perlu diketahui fitur-fiturnya agar bisa dikenali dan bisa dibedakan dari objek yang lain. Fitur-fitur optimal yang bisa diketahui dari suatu objek akan mempermudah dan mempercepat proses indentifikasi objek tersebut. Fitur yang sedikit akan mempermudah menentukan daerah keputusan (*decision regions*) dan sebuah pengklasifikasian lebih mudah untuk dilakukan. Permasalahan pada pemilihan fitur ini adalah pengumpulan fitur-fitur yang akan dipilih dan pemilihan himpunan bagian dari kumpulan fitur-fitur yang tersedia dicari yang paling baik dalam sistem klasifikasi. Dalam hal ini, salah satu kesulitan yang ada adalah bagaimana memilih fitur terbaik dari sekumpulan fitur-fitur yang sangat banyak untuk digunakan dalam pengklasifikasian.

Metode untuk menemukan fitur-fitur yang optimal dari sebuah himpunan fitur-fitur telah banyak dijumpai. Misalnya metode penyelesaian-tunggal (*Single-Solution Methods*) yang dimulai dengan sebuah penyelesaian tunggal (sebuah himpunan bagian dari fitur) dan secara iterasi menambah atau menghilangkan fitur sampai ditemukan kriteria perhentian. Metode ini dibagi menjadi dua, yaitu metode pertama merupakan suatu metode yang diawali dengan himpunan kosong

kemudian ditambahkan fitur satu persatu (*bottom-up* atau *forward*). Metode kedua merupakan suatu metode yang diawali dengan himpunan yang sudah ada fitur-fitur dan fitur-fitur tersebut dihilangkan satu persatu (*top-down* atau *backward*). Algoritma ini tidak menghasilkan hasil yang optimal, karena waktu pencariannya tidak cepat dan membutuhkan banyak perhitungan.

Algoritma *Branch and Bound* mengurangi beban penghitungannya tanpa mengurangi keoptimalannya menggunakan kesamaan sifat (*monotonicity property*) dari fungsi kriteria. Algoritma *Branch and Bound* menghindari penghitungan satu persatu dari himpunan fitur yang ada tanpa mengurangi keoptimalannya dalam pemilihan fitur. Sehingga hal ini dapat menghemat waktu penghitungan dan mengurangi jumlah perhitungan.

1.2 TUJUAN

Tujuan dari tugas akhir ini adalah membuat aplikasi untuk mendapatkan algoritma *Branch and Bound* yang optimal dalam pemilihan fitur.

1.3 PERMASALAHAN

Permasalahan yang diangkat dalam tugas akhir ini adalah:

1. Pemahaman algoritma *Branch and Bound* dasar pada pemilihan fitur.
2. Pemahaman algoritma IBB (*Improved Branch and Bound*) pada pemilihan fitur.
3. Pemahaman algoritma BBPP (*Branch and Bound with Partial Prediction*) pada pemilihan fitur.
4. Pemahaman algoritma FBB (*Fast Branch and Bound*) pada pemilihan fitur.

5. Optimasi yang dilakukan pada algoritma *Branch and Bound* adalah optimasi pada evaluasi fungsi kriteria yang digunakan.

1.4 BATASAN MASALAH

Batasan masalah dalam pengerjaan tugas akhir ini adalah untuk melakukan uji coba digunakan data sintetik dengan fitur-fitur yang telah diurutkan berdasarkan sifat *monotonicity* sebanyak 30 fitur dan dengan fungsi kriteria yang telah ditentukan sebanyak 5 variasi.

1.5 METODOLOGI

Pembuatan tugas akhir ini dilakukan dengan mengikuti metodologi sebagai berikut:

1. Studi literatur

   Mencari dan mempelajari referensi-referensi yang berkaitan dengan seleksi fitur dan algoritma *Branch and Bound.* Pembelajaran ini didapat baik dari buku-buku literatur maupun beberapa artikel di internet. Selain itu dipelajari juga bahasa pemrograman yang akan digunakan, yaitu Matlab 6.5.1

2. Pengembangan Algoritma

   Pada tahap ini dirancang penerapan algoritma yang akan digunakan pada perancangan perangkat lunak yaitu *Branch and Bound* dasar (BBB), *Improved Branch and Bound* (IBB), *Branch and Bound with Partial Prediction* (BBPP), dan *Fast Branch and Bound* (FBB).

3. Perancangan Perangkat Lunak

Pada tahap ini akan dilakukan perancangan terhadap perangkat lunak yang meliputi data yang akan digunakan, proses-proses yang akan dilaksanakan dan penentuan rancangan antar muka berdasarkan studi pustaka yang telah dilakukan.

4. Pembuatan Perangkat Lunak

   Pada tahap ini, perangkat lunak dibuat sesuai dengan desain yang telah dirancang sebelumnya. Data yang digunakan, proses masukan, algoritma dan keluaran didasarkan pada rancangan yang telah dibuat pada tahap sebelumnya.

5. Uji Coba dan Evaluasi Perangkat Lunak

   Pada tahap ini, program yang telah dibuat diuji kebenarannya.

6. Penyusunan Buku tugas akhir

   Tahap ini dilakukan seiring dengan pengerjaan tugas akhir ini. Penulisan dokumentasi meliputi dasar teori dari sistem, desain sistem algoritma yang digunakan, kinerja sistem yang dibangun serta dokumentasi tahap perancangan dan pembuatan.

1.6 SISTEMATIKA PENULISAN

Dalam penulisannya, laporan tugas akhir ini dikelompokkan menjadi tujuh bab, sebagai berikut:

BAB I Pendahuluan

Bab ini menjelaskan tentang latar belakang yang mendasari pembuatan tugas akhir ini, permasalahan, batasan masalah, tujuan dari pembuatan perangkat lunak, metodologi pembuatan serta sistematika penulisan buku tugas akhir.

BAB II Metode Seleksi Fitur Menggunakan Algoritma *Branch and Bound* Dasar dan Algoritma *Improved Branch and Bound*

Bab ini menjelaskan beberapa teori-teori tentang algoritma *Branch and Bound* dan algoritma *Improved Branch and Bound* yang digunakan pada tugas akhir ini. Tahap-tahap dari algoritma *Branch and Bound* dasar dan algortima *Improved Branch and Bound* dijelaskan secara detil pada bab ini. Pada bab ini juga akan diberikan contoh kasus seleksi fitur yang diselesaikan menggunakan algoritma *Branch and Bound* dasar dan algoritma *Improved Branch and Bound.*

BAB III Metode Seleksi Fitur Menggunakan Algoritma *Branch and Bound Partial Prediction* dan Algoritma *Fast Branch and Bound*

Bab ini menjelaskan beberapa teori-teori tentang algoritma *Branch and Bound Partial Prediction* dan algoritma *Fast Branch and Bound* yang digunakan pada tugas akhir ini. Tahap-tahap dari algoritma *Branch and Bound Partial Prediction* dan algortima *Fast Branch and Bound* dijelaskan secara detil pada bab ini. Pada bab ini juga akan diberikan contoh kasus seleksi fitur yang diselesaikan menggunakan algoritma *Branch and Bound Partial Prediction* dan algoritma *Fast Branch and Bound*, serta dapat dilihat perbandingan evaluasi fungsi kriteria yang terjadi pada keempat algoritma *Branch and Bound* pada contoh yang telah dijelaskan.

BAB IV Desain Perangkat Lunak

Bab ini menjelaskan mengenai desain perangkat lunak optimasi pemilihan fitur menggunakan algoritma *Branch and Bound.* Pembahasan ini meliputi lingkungan desain perangkat lunak, masukan dan luaran, desain proses serta antarmuka

aplikasi. Desain perangkat lunak ini menggunakan *flowchart*. *Flowchart* dipergunakan untuk menjelaskan alur proses yang terjadi.

BAB V Implementasi Perangkat Lunak

Bab ini menjelaskan mengenai implementasi optimasi pemilihan fitur menggunakan algoritma *Branch and Bound*. Pembahasan ini meliputi lingkungan implementasi perangkat lunak, serta *pseudocode* program untuk setiap proses yang terjadi pada keempat algoritma *Branch and Bound*.

BAB VI Uji Coba dan Evaluasi

Bab ini menjelaskan hasil uji coba perangkat lunak, dan kemudian dilakukan evaluasi terhadap hasil uji coba tersebut.

BAB VII Kesimpulan

Bab ini berisi kesimpulan dari pembuatan tugas akhir.

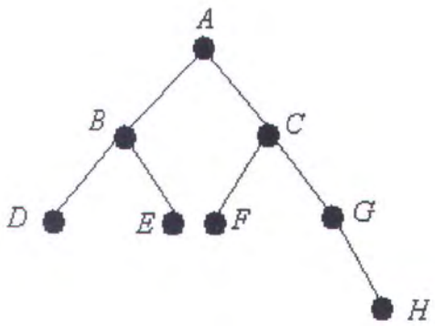
BAB II

METODE SELEKSI FITUR MENGGUNAKAN ALGORITMA *BRANCH & BOUND* DASAR DAN ALGORITMA *IMPROVED BRANCH & BOUND*

Pada bab ini dijelaskan algoritma *Basic Branch & Bound* (BBB) dan algoritma *Improved Branch & Bound* (IBB) yang digunakan dalam seleksi fitur. Seleksi fitur merupakan pemilihan fitur-fitur penting dari himpunan fitur-fitur yang ada. Fitur-fitur penting ini sangat mempengaruhi dalam suatu pengidentifikasian. Oleh karena itu, fitur-fitur yang tidak penting harus dihilangkan agar lebih optimal. Metode seleksi fitur yang dibahas pada bab ini adalah algoritma *Basic Branch & Bound* dan algoritma *Improved Branch & Bound*.

2.1 *TREE* DAN *SEARCH TREE*

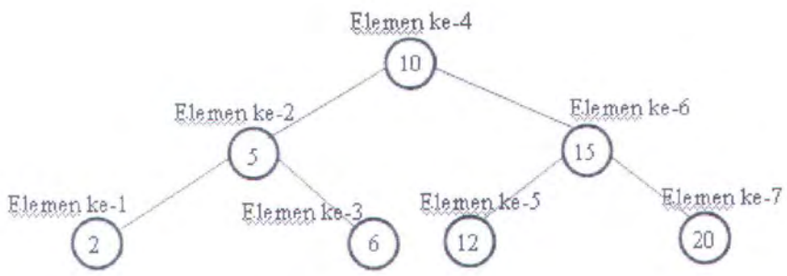
Tree adalah suatu struktur data yang dimulai dari *node root*. *Node-node* dalam *tree* bisa berupa *leaf* atau *internal node*. Sebuah *internal node* bisa memiliki satu *child node* atau lebih. *Internal node* yang memiliki *child node* ini disebut sebagai *parent node*. Semua *child node* dengan *parent node* yang sama disebut *siblings*. *Root* dari sebuah *tree* biasanya diletakkan pada *node* paling atas dan *leaves* diletakkan pada *node* paling bawah. Gambar dari *tree* ini dapat dilihat pada Gambar 2.1.



Gambar 2.1 ***Tree***

Dari Gambar 2.1 dapat dilihat dengan jelas bahwa *node* A merupakan *root* dari *tree*, *node* B merupakan *parent node* dari *node* D dan E, *node* F dan G merupakan *child node* dari *node* C, *node* D, E, F dan H merupakan *leaves*, *node* B, C dan G disebut *internal node*, *node* D dan E *siblings*, dan B, D, E merupakan *subtree*.

Search tree adalah sebuah *tree* dengan sebuah *node* dari tiap *subtree*-nya memiliki nilai yang lebih kecil dari *node subtree* disebelah kanannya. Nilai pada sebuah *node*-nya diantara nilai pada *subtree*-nya dan lebih besar dari nilai *subtree* disebelah kirinya serta lebih kecil dari nilai *subtree* disebelah kanannya. *Binary search tree* merupakan salah satu *search tree* yang membagi sekumpulan data menjadi dua. Misalnya ada sekumpulan data (2,5,6,10,12,15,20), maka elemen tengah dari data tersebut akan dijadikan sebagai *root*, semua elemen yang lebih kecil dari elemen tengah menjadi *subtree* sebelah kiri dan semua elemen yang lebih besar dari elemen tengah menjadi *subtree* sebelah kanan. *Binary tree* dari sekumpulan data tersebut dapat dilihat pada Gambar 2.2.



Gambar 2.2 Binary tree dengan data (2,5,6,10,12,15,20)

Berikut adalah istilah-istilah yang digunakan dalam *tree* dan *search tree* [NIST]:

a. *Subtree* adalah sebuah *tree* yang merupakan *child* dari sebuah node.

b. *Root* merupakan *parent node* yang tidak memiliki *parent*.

c. *Leaf* merupakan akhir dari sebuah *tree*. *Leaf* ini biasanya berada paling bawah dari sebuah *tree* dan tidak memiliki *child node*.

d. *Internal node* merupakan sebuah node dari *tree* yang memiliki satu atau lebih *child node* dan *internal node* ini bukan *root* atau *leaf*.

2.2 ALGORITMA *BRANCH & BOUND* DASAR

Algoritma *Branch & Bound* telah dikembangkan untuk mendapatkan penyelesaian yang optimal pada permasalahan kompleks tanpa adanya perhitungan yang mendalam. Algoritma ini menggunakan sifat *monotonicity* dari fungsi kriteria untuk menyelesaikan pencarian fitur penting. Sifat *monotonicity* digunakan untuk menghilangkan fitur-fitur yang tidak penting dari himpunan fitur yang ada.

Sifat *monotonicity* harus memenuhi persamaan 2.1 dan 2.2. Kedua persamaan ini disebut *monotonicity condition.*

$$\overline{X_1} \supset \overline{X_2} \supset .... \overline{X_j} \quad \text{.....(2.1)}$$

$$J(\overline{X_1}) \geq J(\overline{X_2}) \geq ... J(\overline{X_j}) \quad \text{.....(2.2)}$$

$\overline{X_1}$ merupakan himpunan bagian fitur yang telah dihilangkan satu fitur dari himpunan fitur awal. $J(\overline{X_1})$ merupakan nilai kriteria dari $\overline{X_1}$. *Monotonicity* terjadi bila himpunan $\overline{X_2}$ merupakan himpunan bagian dari $\overline{X_1}$ maka nilai kriteria $\overline{X_1}$ harus lebih besar dari nilai kriteria $\overline{X_2}$.

2.2.1 Prosedur *Branch & Bound* Dasar

Algoritma *Branch & Bound* dasar digunakan untuk memilih sejumlah fitur penting dari himpunan fitur awal dengan menghilangkan fitur-fitur yang tidak penting. Bentuk penyelesaian dari algoritma *Branch & Bound* dasar ini adalah sebuah *tree.*

Prosedur yang digunakan dalam algoritma *Branch & Bound* dasar adalah sebagai berikut [F90]:

1. Jumlah fitur pada himpunan awal dinyatakan dengan D, jumlah fitur yang akan dipilih dinyatakan dengan d dan jumlah fitur yang akan dihilangkan dinyatakan dengan :

   $$\overline{d} = D - d \quad \text{.....(2.3)}$$

2. Himpunan fitur yang akan dibuang dinotasikan dengan

   $$(z_1, ..., z_{\overline{d}}) \quad \text{.....(2.4)}$$

   dengan syarat

$z_1 < z_2 < ... < z_{\bar{d}}$ .....(2.5)

3. Fungsi kriteria $J_{\bar{d}}(z_1,...,z_{\bar{d}})$ merupakan fungsi untuk mendapatkan fitur-fitur yang akan dipilih.

4. Mencari *subset* optimal $(z_1^*,..., z_{\bar{d}}^*)$ dengan cara yaitu,

$$J_{\bar{d}}(z_1^*,...,z_{\bar{d}}^*) = \max_{z_1,...,z_{\bar{d}}} J_{\bar{d}}(z_1,...,z_{\bar{d}}) \quad .....(2.6)$$

5. Fungsi kriteria J harus memenuhi syarat *monotonicity* yaitu,

$$J_1(z_1) \geq J_2(z_1,z_2) \geq ... \geq J_{\bar{d}}(z_1,...,z_{\bar{d}}) \quad .....(2.7)$$

6. x^* merupakan *bound* yang didapat dari nilai max $J_{\bar{d}}(z_1,...,z_{\bar{d}})$ yaitu,

Jika $J_k(z_1,...,z_k) \leq x^*$ untuk $k < \bar{d}$

maka

$$J_{\bar{d}}(z_1,...,z_k,z_{k+1},...,z_{\bar{d}}) \leq J_k(z_1,...,z_k) \leq x^*$$

untuk $\{z_{k+1},...,z_{\bar{d}}\}$ .....(2.8)

Hal ini berarti:

Jika nilai kriteria tiap *node* < x^*, maka tidak optimal dan bisa dilakukan *cu -off.*

2.2.2 Algoritma *Branch & Bound* Dasar

Algoritma *Branch & Bound* dasar ini menggunakan *search tree* untuk menyelesaikan pencarian fitur.

Pertama-tama algoritma dimulai dari *root* sebuah *tree* dan didapatkan *current node* yang disimpan pada *LIST(k).* Kemudian dipilih node yang paling *maximum* sesuai dengan fungsi kriteria yang dipakai (*max* $J_k(z_1,...,z_k)$) untuk

dijadikan sebagai *current node* yang baru, dan algoritma dilanjutkan ke *level* yang lebih tinggi. Hal ini dlakukan terus sampai didapatkan nilai *bound.*

Notasi yang digunakan dalam algoritma *Branch & Bound* ini adalah sebagai berikut [F90]:

a) k menunjukkan level dari *tree.*

b) $LIST(k)$ merupakan himpunan fitur-fitur *candidate* pada *level* ke-k.

c) z merupakan fitur yang akan dibuang.

d) x^* menyimpan nilai *bound.*

Langkah-langkah algoritma *Branch & Bound* dasar adalah sebagai berikut [F90]:

Langkah 1 Inisialisasi:

nilai $x^* = -\sim$, $k = 1$, $z_0 = 0$.

Langkah 2 *Generate successors*:

Inisialisasi $LIST(k)$ yang isinya adalah fitur-fitur $(z_1, ..., z_{k-1})$ dengan cara:

$$LIST(k) = \{z_{k-1} + 1, z_{k-1} + 2, ..., d + k\} \quad \text{.....(2.9)}$$

Langkah 3 Memilih node baru:

- Jika $LIST(k)$ kosong, dilanjutkan ke **Langkah 5**

  jika $LIST(k)$ tidak kosong, maka

  $z_k = m$, dimana $J_k(z_1, ... z_{k-1}, m) = \max_{j \in LIST(k)} J_k(z_k, .. z_{k-1}, j)$.

- m dihapus dari $LIST(k)$.

Langkah 4 *Check Bound* :

- Jika $J_k(z_1, ..., z_k) < x^*$, maka dilanjutkan ke **Langkah 5**.

jika $k = \bar{d}$, maka dilanjutkan ke **Langkah 6.**

Lainnya $k = k+1$ dan kemudian lanjutkan ke **Langkah 2**.

Langkah 5 *Backtrack* ke *level* sebelumnya:

- $k = k-1$
- Jika $k = 0$, maka algoritma telah selesai

  Jika $k \neq 0$, maka dilanjutkan ke **Langkah 3**.

Langkah 6 *Level* terakhir dan *update bound* :

- $x^* = J_{\bar{d}}(z_1,...,z_{\bar{d}})$ disimpan sebagai nilai *bound*
- $(z_1, z_2,..., z_{\bar{d}})$ disimpan sebagai $(z_1^*, z_2^*,..., z_{\bar{d}}^*)$, kemudian dilanjutkan ke **Langkah 5**.

2.2.3 Contoh Algoritma *Branch & Bound* Dasar

Suatu himpunan fitur awal dengan jumlah fiturnya sebanyak 5 fitur akan dipilih 2 fitur saja dengan fungsi kriterianya adalah $J(\bar{x}) = \sum_{\xi_i \in x} i$.

Berdasarkan pada permasalahan diatas, langkah penyelesaian menggunakan algoritma *Branch & Bound* dasar adalah sebagai berikut:

Langkah 1 Inisialisasi awal:

$$D = 5, \quad d = 2, \quad \bar{d} = 5-2 = 3, \quad J(\bar{x}) = \sum_{\xi_i \in x} i, \quad k = 1, \quad x^* = \sim,$$

$$z_0 = \{\xi_1, \xi_2, \xi_3, \xi_4, \xi_5\}$$

Langkah 2 Inisialisasi *LIST*(1)

Kemudian dicari:

- $LIST(1) = \{z_{k-1}+1, z_{k-1}+2, ..., 2+k\}$

  $LIST(1) = \{1,2,3\}$

  kemudian dilanjutkan ke **Langkah 3**.

Langkah 3 Memilih *node* baru:

Telah diketahui:

$k = 1$, $LIST(1) = \{1,2,3\}$, $x^* = \sim$

kemudian di periksa:

- Jika $LIST(1) = \phi$

  Telah diketahui bahwa $LIST(1) = \{1,2,3\}$, maka hal ini tidak memenuhi kondisi yang diminta yaitu $LIST(1) = \phi$.

  Lainnya

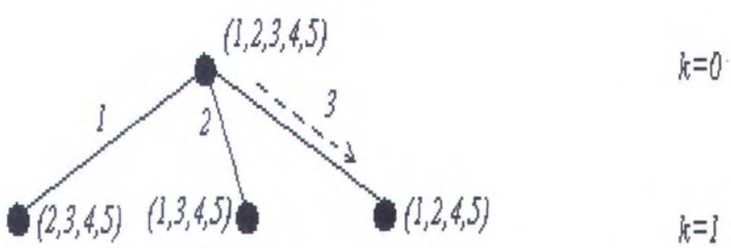
  $z_1 = 3$

- 3 dihapus dari $LIST(1)$, sehingga:

  $LIST(1) = LIST(1) \setminus \{3\}$

  $LIST(1) = \{1,2\}$

dari sini didapat *tree* sebagai berikut:



Gambar 2.3 ***Node-node*** **pada** ***tree level*** **1**

dan kemudian dilanjutkan ke **Langkah 4**.

Langkah 4 *Check Bound* :

Telah diketahui:

$k=1$, $LIST(1)=\{1,2\}$, $z_1=3$, $x^*=\sim$

kemudian diperiksa:

- *if* $J_1(1,2,4,5)<x^*$

  $J_1(1,2,4,5)=12$ dan $x^*=\sim$, sehingga hal ini tidak memenuhi kondisi yang diminta.

- *if* $k=\overline{d}$

  Telah diketahui bahwa $k=1$ dan $\overline{d}=3$, sehingga hal ini tidak memenuhi kondisi yang diminta.

- *Else*

  $k=k+1$

  $k=1+1=2$

  kemudian dilanjutkan ke **Langkah 2**.

Pada langkah ini terjadi perhitungan nilai kriteria, sehingga jumlah evaluasi kriterianya bertambah 1 menjadi 1.

Langkah 2 Inisialisasi $LIST(2)$

Telah diketahui:

$k=2$, $LIST(1)=\{1,2\}$, $z_1=3$, $x^*=\sim$, $J_1(1,2,4,5)=12$

kemudian dicari:

- $LIST(2)=\{z_{k-1}+1, z_{k-1}+2,...,2+k\}$

  $LIST(2)=\{4\}$

  kemudian dilanjutkan ke **Langkah 3**.

Langkah 3 Memilih *node* baru:

Telah diketahui:

$k = 2$, $LIST(1) = \{1,2\}$, $LIST(2) = \{4\}$, $z_1 = 3$, $x^* = \sim$, $J_1(1,2,4,5) = 12$

kemudian di periksa:

- Jika $LIST(2) = \phi$

  Telah diketahui bahwa $LIST(2) = \{4\}$, maka hal ini tidak memenuhi kondisi yang diminta yaitu $LIST(2) = \phi$.

  Lainnya

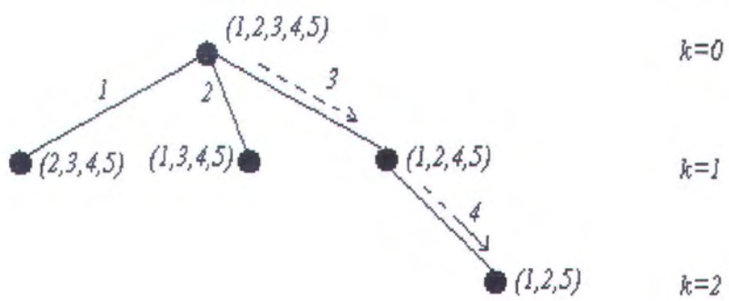
  $z_2 = 4$

- 4 dihapus dari $LIST(2)$, sehingga:

  $LIST(2) = LIST(2) \setminus \{4\}$

  $LIST(2) = \phi$

dari sini didapat *tree* seperti pada Gambar 2.4.



Gambar 2.4 ***Node-node*** **pada** ***tree level*** **2**

dan kemudian dilanjutkan ke **Langkah 4**.

Langkah 4 *Check Bound* :

Telah diketahui:

$k=2$, $LIST(1)=\{1,2\}$, $LIST(2)=\phi$, $z_1=3$, $z_2=4$, $x^*=\sim$,

$J_1(1,2,4,5)=12$

kemudian diperiksa:

- *if* $J_2(1,2,5)<X^*$

  $J_2(1,2,5)=8$ dan $x^*=\sim$, sehingga hal ini tidak memenuhi kondisi yang diminta.

- *if* $k=\overline{d}$

  Telah diketahui bahwa $k=2$ dan $\overline{d}=3$, sehingga hal ini tidak memenuhi kondisi yang diminta.

- *Else*

  $k=k+1$

  $k=2+1=3$

  kemudian dilanjutkan ke **Langkah 2**.

Pada langkah ini terjadi perhitungan nilai kriteria, sehingga jumlah evaluasi kriterianya bertambah 1 menjadi 2.

Langkah 2 Inisialisasi $LIST(3)$

Telah diketahui:

$k=3$, $LIST(1)=\{1,2\}$, $LIST(2)=\phi$, $z_1=3$, $z_2=4$, $x^*=\sim$,

$J_1(1,2,4,5)=12$, $J_2(1,2,5)=8$

kemudian dicari:

- $LIST(3)=\{z_2+1, z_2+2,\ldots,2+3\}$

  $LIST(3)=\{5\}$

  kemudian dilanjutkan ke **Langkah 3**.

Langkah 3 Memilih *node* baru:

Telah diketahui:

$k=3$, $LIST(1)=\{1,2\}$, $LIST(2)=\phi$, $LIST(3)=\{5\}$, $z_1=3$, $z_2=4$, $x^*=\sim$,

$J_1(1,2,4,5)=12$, $J_2(1,2,5)=8$

kemudian di periksa:

- Jika $LIST(3)=\phi$

  Telah diketahui bahwa $LIST(3)=\{5\}$, maka hal ini tidak memenuhi kondisi yang diminta yaitu $LIST(3)=\phi$.

  Lainnya

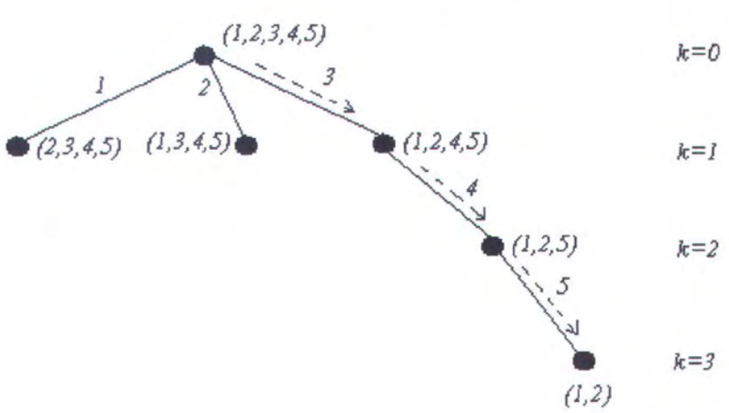
  $z_3=5$

- 5 dihapus dari $LIST(3)$, sehingga:

  $LIST(3)=LIST(3)\setminus\{3\}$

  $LIST(3)=\phi$

dari sini didapat *tree* seperti pada Gambar 2.5.



Gambar 2.5 ***Node-node*** **pada** ***tree level*** **3**

dan kemudian dilanjutkan ke **Langkah 4**.

Langkah 4 *Check Bound* :

Telah diketahui:

$k=3$, $LIST(1)=\{1,2\}$, $LIST(2)=\phi$, $LIST(3)=\phi$, $z_1=3$, $z_2=4$, $z_3=5$,

$x^*=\sim$, $J_1(1,2,4,5)=12$, $J_2(1,2,5)=8$

kemudian diperiksa:

- *if* $J_3(1,2)<X^*$

  $J_3(1,2)=3$ dan $x^*=\sim$, sehingga hal ini tidak memenuhi kondisi yang diminta.

- *if* $k=\overline{d}$

  Telah diketahui bahwa $k=3$ dan $\overline{d}=3$, sehingga memenuhi kondisi yang diminta dan dilanjutkan ke **Langkah 6**.

Pada langkah ini terjadi perhitungan nilai kriteria, sehingga jumlah evaluasi kriterianya bertambah 1 menjadi 3.

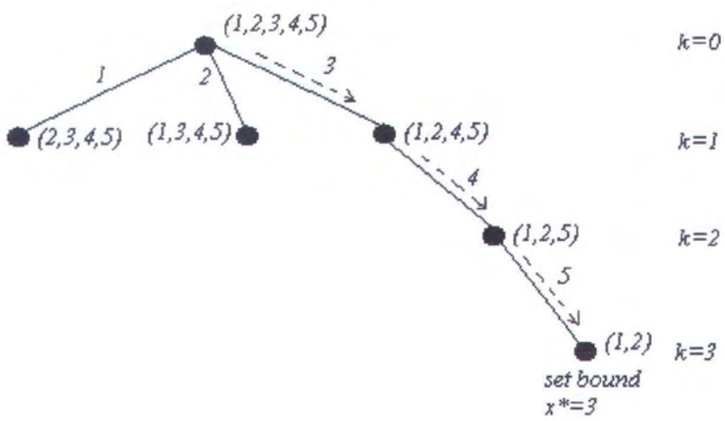
Langkah 6 *Level* terakhir dan *update bound* :

- $x^*=J_3(1,2)$

  $x^*=3$

- $(z_1^*,z_2^*,z_3^*)=(3,4,5)$

  Sehingga didapatkan nilai *bound* adalah 3 seperti pada Gambar 2.6.



Gambar 2.6 *Bound* telah ditemukan pada *node level* 3

dan kemudian dilanjutkan ke **Langkah 5**.

Langkah 5 *Backtrack* ke *level* sebelumnya :

Telah diketahui:

$k=3$, $LIST(1)=\{1,2\}$, $LIST(2)=\phi$, $LIST(3)=\phi$, $z_1=3$, $z_2=4$, $z_3=5$,

$x^*=3$, $J_1(1,2,4,5)=12$, $J_2(1,2,5)=8$, $(z_1^*, z_2^*, z_3^*)=(3,4,5)$

melakukan *bactracking*:

- $k=k-1$

  $k=3-1$

  $k=2$

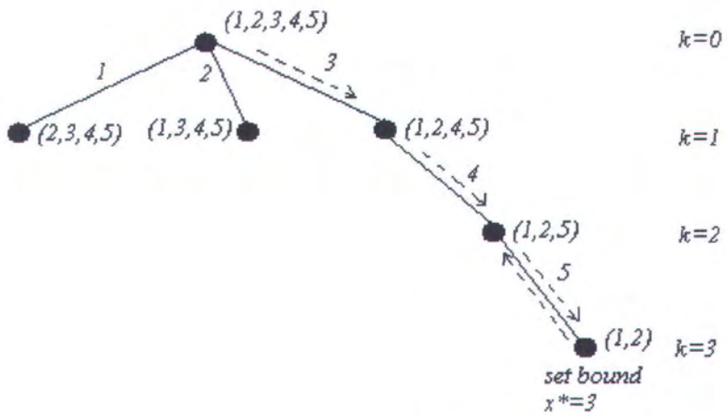
  - *if* $k=0$

    karena $k=2$, maka tidak memenuhi kondisi yang diminta

  - *else*

    dilanjutkan ke **Langkah 3**.

Penelusuran *tree* sekarang kembali ke *level* 2 seperti ditunjukkan oleh gambar 2.7 berikut



Gambar 2.7 Penelusuran *tree* kembali ke *level 2*

Langkah 3 Memilih *node* baru:

Telah diketahui:

$k=2$, $LIST(1)=\{1,2\}$, $LIST(2)=\phi$, $LIST(3)=\phi$, $z_1=3$, $z_2=4$, $z_3=5$,

$x^*=3$, $J_1(1,2,4,5)=12$, $J_2(1,2,5)=8$, $(z_1^*,z_2^*,z_3^*)=(3,4,5)$

kemudian di periksa:

- Jika $LIST(2)=\phi$

  Telah diketahui bahwa $LIST(2)=\phi$, maka memenuhi kondisi yang diminta dan dilanjutkan ke **Langkah 5**.

Langkah 5 *Backtrack* ke *level* sebelumnya:

Telah diketahui:

$k=2$, $LIST(1)=\{1,2\}$, $LIST(2)=\phi$, $LIST(3)=\phi$, $z_1=3$, $z_2=4$, $z_3=5$,

$x^*=3$, $J_1(1,2,4,5)=12$, $J_2(1,2,5)=8$, $(z_1^*,z_2^*,z_3^*)=(3,4,5)$

melakukan *bactracking*:

- $k = k - 1$

  $k = 2 - 1$

  $k = 1$

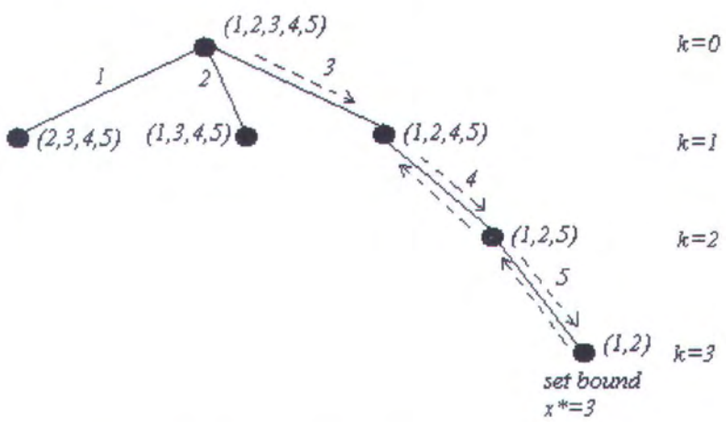
  - *if* $k = 0$

    karena $k = 1$, maka tidak memenuhi kondisi yang diminta

  - *else*

    Dilanjutkan ke **Langkah 3**.

  Penelusuran *tree* sekarang kembali ke *level* 1 seperti ditunjukkan oleh gambar 2.8 berikut



Gambar 2.8 Penelusuran *tree* kembali ke *level* 1

Langkah 3 Memilih *node* baru:

Telah diketahui:

$k = 1$, $LIST(1) = \{1,2\}$, $LIST(2) = \phi$, $LIST(3) = \phi$, $z_1 = 3$, $z_2 = 4$, $z_3 = 5$,

$x^* = 3$, $J_1(1,2,4,5) = 12$, $J_2(1,2,5) = 8$, $(z_1^*, z_2^*, z_3^*) = (3,4,5)$

kemudian di periksa:

- Jika $LIST(1) = \phi$

  Telah diketahui bahwa $LIST(1) = \{1,2\}$, maka hal ini tidak memenuhi kondisi yang diminta yaitu $LIST(1) = \phi$.

  Lainnya

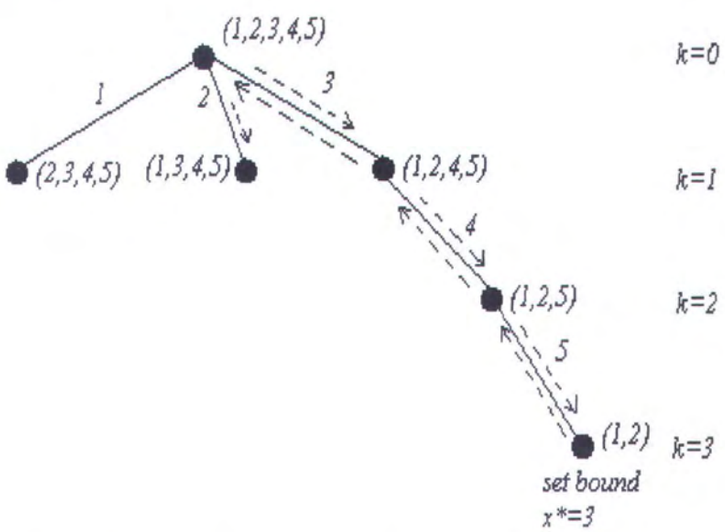
  $z_1 = 2$

- 2 dihapus dari $LIST(1)$, sehingga:

  $LIST(1) = LIST(1) \setminus \{2\}$

  $LIST(1) = \{1\}$

Pada kondisi ini, penelusuran *tree* dilanjutkan pada *node* sebelah kirinya. Pada Gambar 2.9 menunjukkan perpindahan ke *node* sebelah kirinya.



Gambar 2.9 ***Node-node*** **pada** ***tree*** **level 1 setelah dilakukan** ***backtracking***

dan kemudian dilanjutkan ke **Langkah 4**.

Langkah 4 *Check Bound* :

Telah diketahui:

$k=1$, $LIST(1)=\{1\}$, $LIST(2)=\phi$, $LIST(3)=\phi$, $z_1=2$, $z_2=4$, $z_3=5$,

$x^*=3$, $(z_1^*, z_2^*, z_3^*)=(3,4,5)$

kemudian diperiksa:

- *if* $J_1(1,3,4,5)<X^*$

  $J_1(1,3,4,5)=13$ dan $x^*=3$, sehingga hal ini tidak memenuhi kondisi yang diminta.

- *if* $k=\overline{d}$

  Telah diketahui bahwa $k=1$ dan $\overline{d}=3$, sehingga hal ini tidak memenuhi kondisi yang diminta.

- *Else*

  $k=k+1$

  $k=1+1=2$

  kemudian diilanjutkan ke **Langkah 2**.

Pada langkah ini terjadi perhitungan nilai kriteria, sehingga jumlah evaluasi kriterianya bertambah 1 menjadi 4.

Langkah 2 Inisialisasi $LIST(2)$

Telah diketahui:

$k=2$, $LIST(1)=\{1\}$, $LIST(2)=\phi$, $LIST(3)=\phi$, $z_1=2$, $z_2=4$, $z_3=5$,

$x^*=3$, $(z_1^*, z_2^*, z_3^*)=(3,4,5)$

kemudian dicari:

- $LIST(2) = \{z_1 + 1, z_1 + 2, ..., 2 + 2\}$

  $LIST(2) = \{3,4\}$

  kemudian dilanjutkan ke **Langkah 3**.

Langkah 3 Memilih *node* baru:

Telah diketahui:

$k = 2$, $LIST(1) = \{1\}$, $LIST(2) = \{3,4\}$, $LIST(3) = \phi$, $z_1 = 2$, $z_2 = 4$,

$z_3 = 5$, $x^* = 3$, $(z_1^*, z_2^*, z_3^*) = (3,4,5)$

kemudian di periksa:

- Jika $LIST(2) = \phi$

  Telah diketahui bahwa $LIST(2) = \{3,4\}$, maka hal ini tidak memenuhi kondisi yang diminta yaitu $LIST(2) = \phi$.

  Lainnya

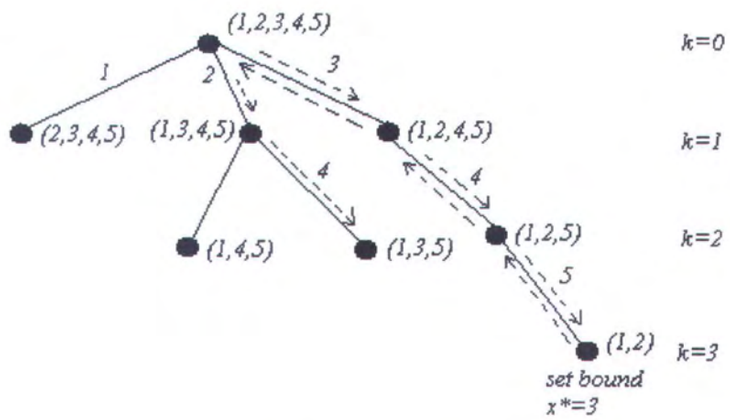
  $z_2 = 4$

- 4 dihapus dari $LIST(2)$, sehingga:

  $LIST(2) = LIST(2) \setminus \{4\}$

  $LIST(2) = \{3\}$

dari sini didapat *tree* seperti pada Gambar 2.10.



Gambar 2.10 ***Node-node*** **pada** ***tree level*** **2**

dan kemudian dilanjutkan ke **Langkah 4**.

Langkah 4 *Check Bound* :

Telah diketahui:

$k = 2$, $LIST(1) = \{1\}$, $LIST(2) = \{3\}$, $LIST(3) = \phi$, $z_1 = 2$, $z_2 = 4$, $z_3 = 5$,

$x^* = 3$, $(z_1^*, z_2^*, z_3^*) = (3,4,5)$

kemudian diperiksa:

- *if* $J_2(1,3,5) < X^*$

  $J_2(1,3,5) = 9$ dan $x^* = 3$, sehingga hal ini tidak memenuhi kondisi yang diminta.

- *if* $k = \overline{d}$

  Telah diketahui bahwa $k = 2$ dan $\overline{d} = 3$, sehingga hal ini tidak memenuhi kondisi yang diminta.

- *Else*

  $k = k + 1$

  $k = 2 + 1 = 3$

kemudian dilanjutkan ke **Langkah 2**.

Pada langkah ini terjadi perhitungan nilai kriteria, sehingga jumlah evaluasi kriterianya bertambah 1 menjadi 5.

Langkah 2 Inisialisasi $LIST(3)$

Telah diketahui:

$k=3$, $LIST(1)=\{1\}$, $LIST(2)=\{3\}$, $LIST(3)=\phi$, $z_1=2$, $z_2=4$, $z_3=5$,

$x^*=3$, $(z_1^*, z_2^*, z_3^*)=(3,4,5)$

kemudian dicari:

- $LIST(3)=\{z_2+1, z_2+2, \ldots, 2+3\}$

  $LIST(3)=\{5\}$

kemudian dilanjutkan ke **Langkah 3**.

Langkah 3 Memilih *node* baru:

Telah diketahui:

$k=3$, $LIST(1)=\{1\}$, $LIST(2)=\{3\}$, $LIST(3)=\{5\}$, $z_1=2$, $z_2=4$, $z_3=5$,

$x^*=3$, $J_1(1,2,4,5)=12$, $J_2(1,2,5)=8$, $(z_1^*, z_2^*, z_3^*)=(3,4,5)$

kemudian di periksa:

- Jika $LIST(3)=\phi$

  Telah diketahui bahwa $LIST(3)=\{5\}$, maka hal ini tidak memenuhi kondisi yang diminta yaitu $LIST(3)=\phi$.

  Lainnya

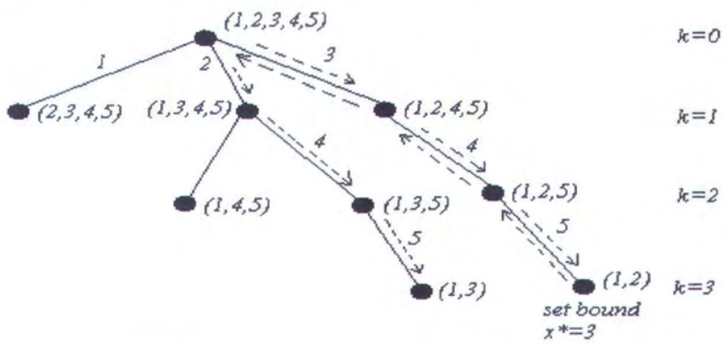
  $z_3=5$

- 5 dihapus dari $LIST(3)$, sehingga:

$LIST(3) = LIST(3) \setminus \{5\}$

$LIST(3) = \phi$

dari sini didapat *tree* seperti pada Gambar 2.11.



Gambar 2.11 ***Node-node*** **pada** ***tree level*** **3**

dilanjutkan ke **Langkah 4**.

Langkah 4 *Check Bound* :

Telah diketahui:

$k = 3$, $LIST(1) = \{1\}$, $LIST(2) = \{3\}$, $LIST(3) = \phi$, $z_1 = 2$, $z_2 = 4$, $z_3 = 5$,

$x^* = 3$, $(z_1^*, z_2^*, z_3^*) = (3,4,5)$

kemudian diperiksa:

- *if* $J_3(1,3) < X^*$

  $J_3(1,3) = 4$ dan $x^* = 3$, sehingga hal ini tidak memenuhi kondisi yang diminta.

- *if* $k = \overline{d}$

  Telah diketahui bahwa $k = 3$ dan $\overline{d} = 3$, sehingga memenuhi kondisi yang diminta dan dilanjutkan ke **Langkah 6**.

Pada langkah ini terjadi perhitungan nilai kriteria, sehingga jumlah evaluasi kriterianya bertambah 1 menjadi 6.

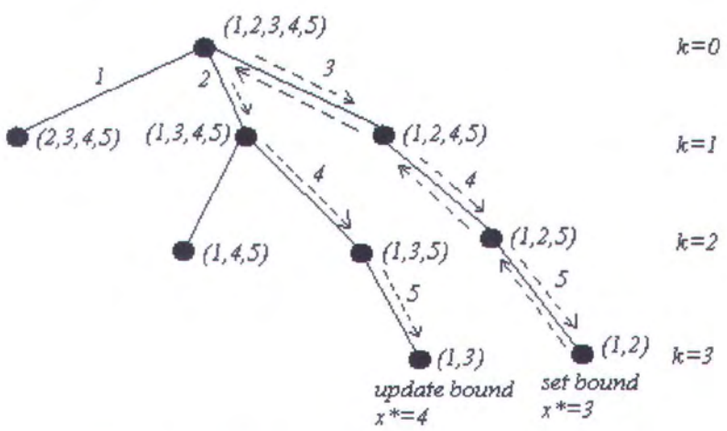
Langkah 6 *Level* terakhir dan *update bound* :

- $x^* = J_3(1,3)$

  $x^* = 4$

- $(z_1^*, z_2^*, z_3^*) = (2,4,5)$

  Sehingga didapatkan nilai *bound* adalah 4 seperti pada Gambar 2.12.



Gambar 2.12 *Bound* diupdate pada *node level* 3

dan kemudian dilanjutkan ke **Langkah 5**.

Langkah 5 *Backtrack* ke *level* sebelumnya:

Telah diketahui:

$k = 3$, $LIST(1) = \{1\}$, $LIST(2) = \{3\}$, $LIST(3) = \phi$, $z_1 = 2$, $z_2 = 4$, $z_3 = 5$,

$x^* = 4$, $(z_1^*, z_2^*, z_3^*) = (3,4,5)$

melakukan *bactracking*:

- $k = k - 1$

$k = 3 - 1$

$k = 2$

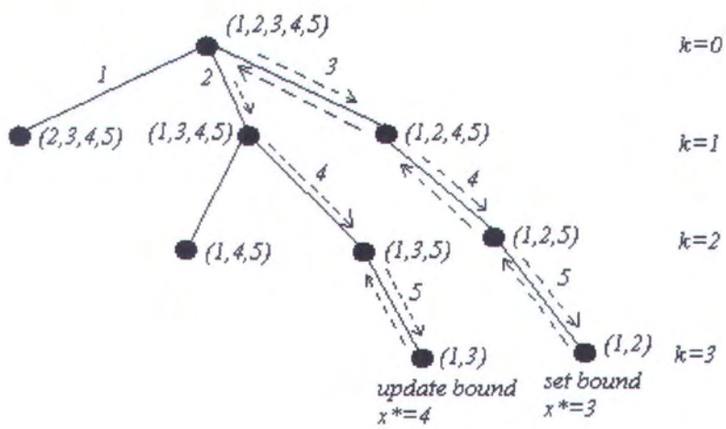
- *if* $k = 0$

  karena $k = 2$, maka tidak memenuhi kondisi yang diminta

- *else*

  Dilanjutkan ke **Langkah 3**.

Penelusuran *tree* sekarang kembali ke *level* 2 seperti ditunjukkan oleh gambar 2.13 berikut



Gambar 2.13 Penelusuran *tree* kembali ke *level 2*

Langkah 3 Memilih *node* baru:

Telah diketahui:

$k = 2$, $LIST(1) = \{1\}$, $LIST(2) = \{3\}$, $LIST(3) = \phi$, $z_1 = 2$, $z_2 = 4$, $z_3 = 5$,

$x^* = 4$, $(z_1^*, z_2^*, z_3^*) = (3,4,5)$

kemudian di periksa:

- Jika $LIST(2) = \phi$

Telah diketahui bahwa $LIST(2)=\{3\}$, maka hal ini tidak memenuhi kondisi yang diminta yaitu $LIST(2)=\phi$.

Lainnya

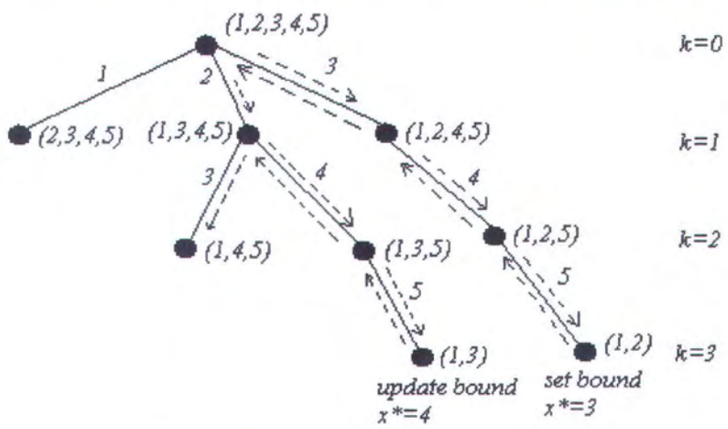
$$z_2 = 3$$

- 3 dihapus dari $LIST(2)$, sehingga:

$$LIST(2) = LIST(2) \setminus \{3\}$$

$$LIST(2) = \phi$$

dari sini didapat *tree* seperti pada Gambar 2.14.



Gambar 2.14 ***Node-node*** **pada** ***tree level*** **2**

dan kemudian dilanjutkan ke **Langkah 4**.

Langkah 4 *Check Bound* :

Telah diketahui:

$k=2$, $LIST(1)=\{1\}$, $LIST(2)=\phi$, $LIST(3)=\phi$, $z_1=2$, $z_2=3$, $z_3=5$,

$x^*=4$, $(z_1^*, z_2^*, z_3^*)=(3,4,5)$

kemudian diperiksa:

- *if* $J_2(1,4,5) < X^*$

$J_2(1,4,5)=10$ dan $x^*=4$, sehingga hal ini tidak memenuhi kondisi yang diminta.

- *if* $k=\bar{d}$

  Telah diketahui bahwa $k=2$ dan $\bar{d}=3$, sehingga hal ini tidak memenuhi kondisi yang diminta.

- *Else*

  $k=k+1$

  $k=2+1=3$

  kemudian dilanjutkan ke **Langkah 2**.

Pada langkah ini terjadi perhitungan nilai kriteria, sehingga jumlah evaluasi kriterianya bertambah 1 menjadi 7.

Langkah 2 Inisialisasi $LIST(3)$

Telah diketahui:

$k=3$, $LIST(1)=\{1\}$, $LIST(2)=\phi$, $LIST(3)=\phi$, $z_1=2$, $z_2=3$, $z_3=5$,

$x^*=4$, $(z_1^*, z_2^*, z_3^*)=(3,4,5)$

kemudian dicari:

- $LIST(3)=\{z_2+1, z_2+2,\ldots,2+3\}$

  $LIST(3)=\{4,5\}$

dan kemudian dilanjutkan ke **Langkah 3**.

Langkah 3 Memilih *node* baru:

Telah diketahui:

$k=3$, $LIST(1)=\{1\}$, $LIST(2)=\phi$, $LIST(3)=\{4,5\}$, $z_1=2$, $z_2=3$, $z_3=5$,

$x^*=4$, $(z_1^*, z_2^*, z_3^*)=(3,4,5)$

kemudian di periksa:

- Jika $LIST(3) = \phi$

  Telah diketahui bahwa $LIST(3) = \{4,5\}$, maka hal ini tidak memenuhi kondisi yang diminta yaitu $LIST(3) = \phi$.

  Lainnya

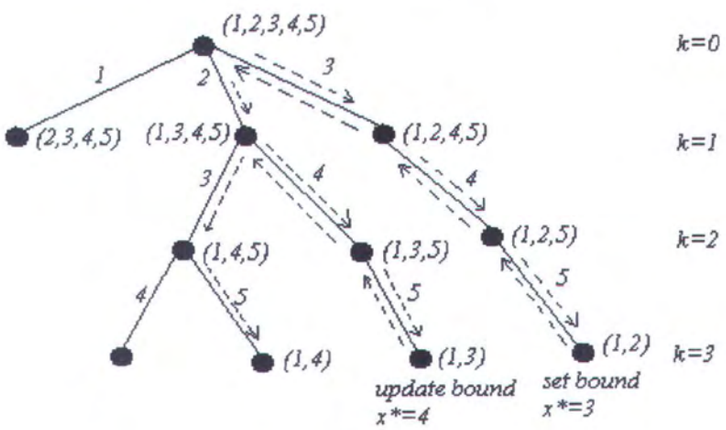
  $$z_3 = 5$$

- 5 dihapus dari $LIST(3)$, sehingga:

  $LIST(3) = LIST(3) \setminus \{5\}$

  $LIST(3) = \{4\}$

dari sini didapat *tree* seperti pada Gambar 2.15.



Gambar 2.15 ***Node-node*** **pada** ***tree level*** **3**

dan kemudian dilanjutkan ke **Langkah 4**.

Langkah 4 *Check Bound* :

Telah diketahui:

$k = 3$, $LIST(1) = \{1\}$, $LIST(2) = \phi$, $LIST(3) = \{4\}$, $z_1 = 2$, $z_2 = 3$, $z_3 = 5$,

$x^* = 4$, $(z_1^*, z_2^*, z_3^*) = (3,4,5)$

kemudian diperiksa:

- *if* $J_3(1,4) < X^*$

  $J_3(1,4) = 5$ dan $x^* = 4$, sehingga hal ini tidak memenuhi kondisi yang diminta.

- *if* $k = \overline{d}$

  Telah diketahui bahwa $k = 3$ dan $\overline{d} = 3$, sehingga memenuhi kondisi yang diminta dan dilanjutkan ke **Langkah 6**.

Pada langkah ini terjadi perhitungan nilai kriteria, sehingga jumlah evaluasi kriterianya bertambah 1 menjadi 8.

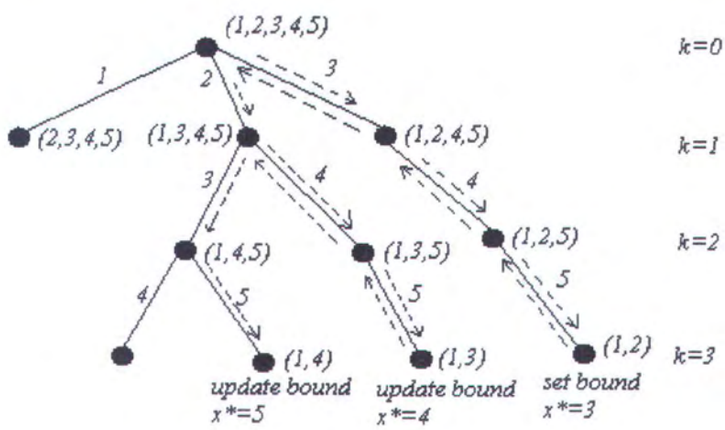
Langkah 6 *Level* terakhir dan *update bound* :

- $x^* = J_3(1,4)$

  $x^* = 5$

- $(z_1^*, z_2^*, z_3^*) = (2,3,5)$

  Sehingga didapatkan nilai *bound* adalah 5 seperti pada Gambar 2.16.



Gambar 2.16 ***Bound*** **diupdate pada** ***node level*** **3**

Dilanjutkan ke **Langkah 5**.

Langkah 5 *Backtrack* ke *level* sebelumnya:

Telah diketahui:

$k=3$, $LIST(1)=\{1\}$, $LIST(2)=\phi$, $LIST(3)=\{4\}$, $z_1=2$, $z_2=3$, $z_3=5$,

$x^*=5$, $(z_1^*, z_2^*, z_3^*)=(2,3,5)$

melakukan *bactracking*:

- $k=k-1$

  $k=3-1$

  $k=2$

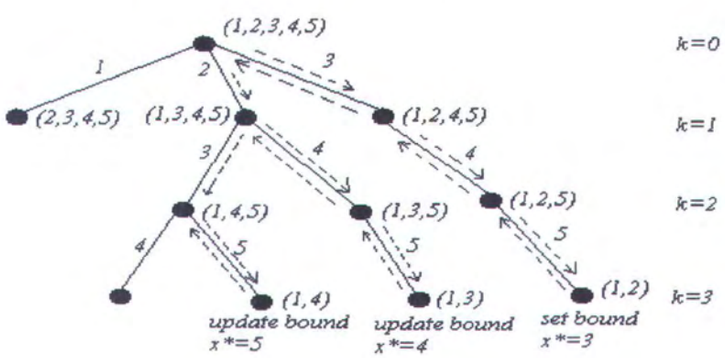
  - *if* $k=0$

    karena $k=2$, maka tidak memenuhi kondisi yang diminta

  - *else*

    Dilanjutkan ke **Langkah 3**.

Penelusuran *tree* sekarang kembali ke *level* 2 seperti ditunjukkan oleh gambar 2.17 berikut



Gambar 2.17 Penelusuran *tree* kembali ke *level 2*

Langkah 3 Memilih *node* baru:

Telah diketahui:

$k = 2$, $LIST(1) = \{1\}$, $LIST(2) = \phi$, $LIST(3) = \{4\}$, $z_1 = 2$, $z_2 = 3$, $z_3 = 5$,

$x^* = 5$, $(z_1^*, z_2^*, z_3^*) = (2,3,5)$

kemudian di periksa:

- Jika $LIST(3) = \phi$

  Telah diketahui bahwa $LIST(3) = \{4\}$, maka hal ini tidak memenuhi kondisi yang diminta yaitu $LIST(3) = \phi$.

  Lainnya

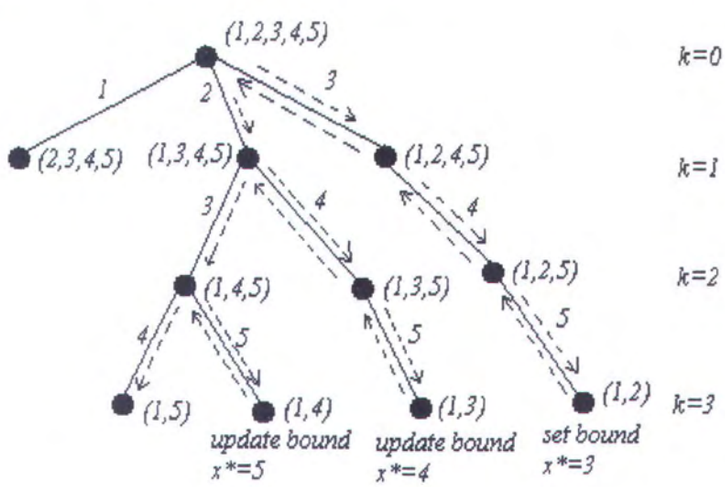
  $z_3 = 4$

- 4 dihapus dari $LIST(3)$, sehingga:

  $LIST(3) = LIST(3) \setminus \{4\}$

  $LIST(3) = \phi$

dari sini didapat *tree* seperti pada Gambar 2.18.



Gambar 2.18 ***Node-node*** **pada** ***tree level*** **3**

dan kemudian dilanjutkan ke **Langkah 4**.

Langkah 4 *Check Bound* :

Telah diketahui:

$k=2$, $LIST(1)=\{1\}$, $LIST(2)=\phi$, $LIST(3)=\phi$, $z_1=2$, $z_2=3$, $z_3=4$,

$x^*=5$, $(z_1^*, z_2^*, z_3^*)=(2,3,5)$

kemudian diperiksa:

- *if* $J_3(1,5)<X*$

  $J_3(1,5)=6$ dan $x^*=5$, sehingga hal ini tidak memenuhi kondisi yang diminta.

- *if* $k=\overline{d}$

  Telah diketahui bahwa $k=3$ dan $\overline{d}=3$, sehingga memenuhi kondisi yang diminta dan dilanjutkan ke **Langkah 6**.

Pada langkah ini terjadi perhitungan nilai kriteria, sehingga jumlah evaluasi kriterianya bertambah 1 menjadi 9.

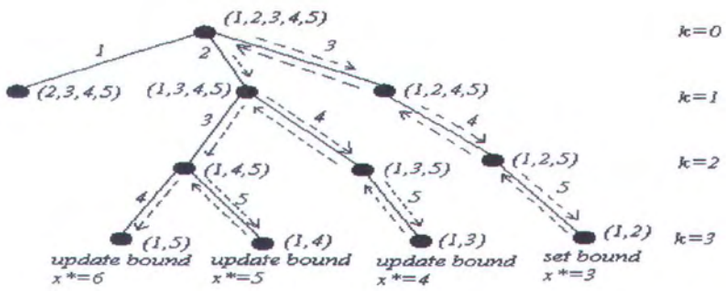
Langkah 6 *Level* terakhir dan *update bound* :

- $x^*=J_3(1,5)$

  $x^*=6$

- $(z_1^*, z_2^*, z_3^*)=(2,3,4)$

  Sehingga didapatkan nilai *bound* adalah 6 seperti pada Gambar 2.19.



Gambar 2.19 *Bound* diupdate pada *node level* 3

dan kemudian dilanjutkan ke **Langkah 5**.

Langkah 5 *Backtrack* ke *level* sebelumnya:

Telah diketahui:

$k=3$, $LIST(1)=\{1\}$, $LIST(2)=\phi$, $LIST(3)=\phi$, $z_1=2$, $z_2=3$, $z_3=4$,

$x^*=6$, $(z_1^*, z_2^*, z_3^*)=(2,3,5)$

melakukan *bactracking*:

- $k=k-1$

  $k=3-1$

  $k=2$

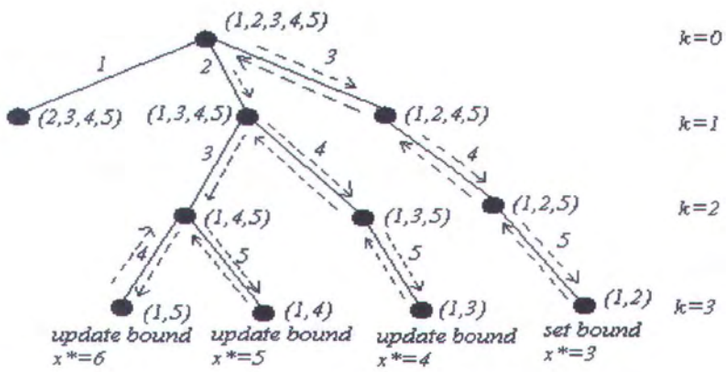
  - *if* $k=0$

    karena $k=2$, maka tidak memenuhi kondisi yang diminta

  - *else*

    Dilanjutkan ke **Langkah 3**.

  Penelusuran *tree* sekarang kembali ke *level* 2 seperti ditunjukkan oleh gambar 2.20 berikut



Gambar 2.20 Penelusuran *tree* kembali ke *level 2*

Langkah 3 Memilih *node* baru:

Telah diketahui:

$k=2$, $LIST(1)=\{1\}$, $LIST(2)=\phi$, $LIST(3)=\phi$, $z_1=2$, $z_2=3$, $z_3=4$,

$x^*=6$, $(z_1^*, z_2^*, z_3^*)=(2,3,5)$

kemudian di periksa:

- Jika $LIST(2)=\phi$

  Telah diketahui bahwa $LIST(2)=\phi$, maka memenuhi kondisi yang diminta dan dilanjutkan ke **Langkah 5**.

Langkah 5 *Backtrack to lower level:*

Telah diketahui:

$k=2$, $LIST(1)=\{1\}$, $LIST(2)=\phi$, $LIST(3)=\phi$, $z_1=2$, $z_2=3$, $z_3=4$,

$x^*=6$, $(z_1^*, z_2^*, z_3^*)=(2,3,5)$

melakukan *bactracking*:

- $k=k-1$

  $k=2-1$

  $k=1$

- *if* $k = 0$

  karena $k = 1$, maka tidak memenuhi kondisi yang diminta

- *else*

  Dilanjutkan ke **Langkah 3**.

Langkah 3 Memilih *node* baru:

Telah diketahui:

$k = 1$, $LIST(1) = \{1\}$, $LIST(2) = \phi$, $LIST(3) = \phi$, $z_1 = 2$, $z_2 = 3$, $z_3 = 4$,

$x^* = 6$, $(z_1^*, z_2^*, z_3^*) = (2,3,5)$

kemudian di periksa:

- Jika $LIST(3) = \phi$

  Telah diketahui bahwa $LIST(3) = \{4\}$, maka hal ini tidak memenuhi kondisi yang diminta yaitu $LIST(3) = \phi$.

  Lainnya

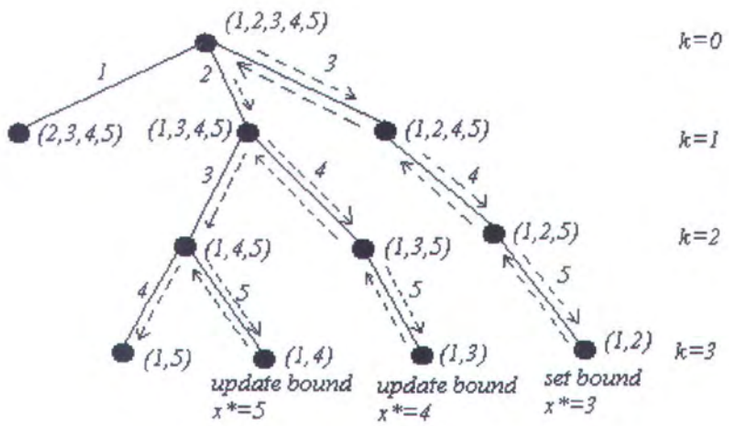
  $z_3 = 4$

- 4 dihapus dari $LIST(3)$, sehingga:

  $LIST(3) = LIST(3) \setminus \{4\}$

  $LIST(3) = \phi$

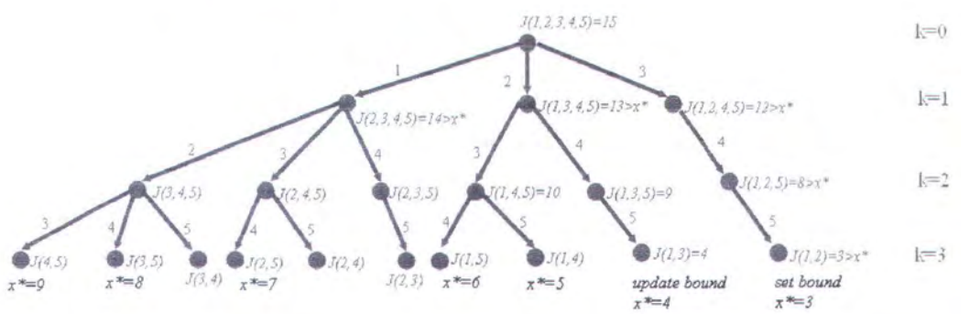
dari sini didapat *tree* seperti pada Gambar 2.21.



Gambar 2.21 ***Node-node*** **pada** ***tree level*** **3**

dan kemudian dilanjutkan ke **Langkah 4**. Demikian seterusnya sampai algoritma selesai.

Hasil akhir dari penelusuran *tree* pada algoritma ini dapat dilihat pada Gambar 2.22.



Gambar 2.22 Hasil akhir penelusuran ***tree*** **pada algoritma BBB**

Dari penelusuran *tree* yang telah dilakukan, didapatkan niali *bound* adalah 9 dan fitur yang dipilih adalah fitur 4 dan 5. Jumlah evaluasi kriteria adalah jumlah setiap terjadi perhitungan nilai kriteria. Jumlah evaluasi kriteria yang terjadi pada algoritma BBB ini secara keseluruhan sebanyak 19.

2.3 ALGORITMA *IMPROVED BRANCH & BOUND*

Algoritma *Improved Branch & Bound* dikembangkan dari algoritma dasarnya yaitu algotirma *Branch & Bound.* Algoritma *Improved Branch & Bound* ini mengurutkan *node-node* fitur pada tiap *level tree* secara menurun. Algoritma *Improved Branch & Bound* bertujuan menempatkan fitur buruk pada *edge* sebelah kanan dari *tree* dan fitur baik pada *edge* sebelah kiri dari *tree.*

Notasi-notasi yang digunakan dalam algoritma *Improved Branch & Bound* adalah sebagai berikut [SPK04]:

a) k : *level* dari *tree* (*root* dinotasikan dengan k=0)

b) $\overline{X_k} = \{\xi_j | j = 1,2,...,D-k\}$: himpunan bagian kandidat fitur pada *level tree* ke-k

c) q_k: jumlah keturunan dari *current node* (pada *level tree* yang berhubungan)

d) $Q_k = \{Q_{k,1}, Q_{k,2}, ..., Q_{k,q_k}\}$: himpunan terurut dari fitur-fitur yang ditandai untuk menjadi *edge* panutan pada keturunan dari *current node* (kandidat himpunan bagian $\overline{X_{k+1}}$ didefinisikan oleh fitur $Q_{k,i}$ untuk i = 1,..., q_k)

e) $J_k = [J_{k,1}, J_{k,2}, ..., J_{k,q_k}]^T$: vektor dari nilai kriteria yang sesuai untuk keturunan dari node pada *level tree* yang berhubungan ($J_{k,i} = J(\overline{X_k} \setminus \{Q_{k,i}\})$ untuk i = 1, ..., q_k)

f) $\Psi = \{\psi_j | j = 1,2,...,r\}$: himpunan kontrol dari sejumlah r fitur yang masih ada yang digunakan untuk pembentukan *search tree*, contohnya, untuk

membentuk himpunan Q_k, himpunan Ψ digunakan untuk memelihara bentuk *search tree*.

g) $X = \{x_j \mid j = 1,2,\ldots,d\}$: himpunan bagian dari fitur terbaik saat ini.

h) X^* : *bound* saat ini (nilai kriteria yang sesuai untuk X)

Langkah-langkah algoritma IBB adalah sebagai berikut [SPK04]:

Inisialisasi:

$k = 0$, $\overline{\chi_0} = Y$, $\Psi = Y$, $r = D$, dan x^* adalah nilai paling kecil yang mungkin

Langkah 1

Pemilihan turunan *node* dari *current node* untuk membentuk *level tree* yang berurutan : pertama, nilai q_k diset dengan

$q_k = r - (D - d - k - 1)$ ....(2.12)

kemudian dibuat himpunan Q_k dan vector J_k sebagai berikut:

Semua fitur $\psi_j \in \Psi, j = 1,\ldots,r$ diurutkan dengan syarat:

$J(\overline{X_k} \setminus \{\psi_{j1}\}) \leq J(\overline{X_k} \setminus \{\psi_{j2}\}) \leq \ldots \leq J(\overline{X_k} \setminus \{\psi_{jr}\})$ ....(2.13)

kemudian dipilih fitur sebanyak q_k dari himpunan fitur pada Ψ, yaitu dengan cara:

$Q_{k,i} = \psi_{j_i}$ untuk i = 1,..., q_k ....(2.14)

$J_{k,i} = J(\overline{X_k} \setminus \{\psi_{j_i}\})$ untuk i = 1,..., q_k ....(2.15)

Untuk menghindari evaluasi ganda, maka fitur ψ_{j_i} dihilangkan dari Ψ, yaitu :

$\Psi = \Psi \setminus Q_k$ ....(2.16)

dan

$r = r - q_k$ .....(2.17).

Langkah 2

node turunan paling kanan (dihubungkan dengan $Q_{k,qk}$ – edge) diperiksa:

If $q_k = 0$, semua turunan telah diuji dan langsung ke **Langkah 4** (*backtracking*)

If $J_{k,q_k} < x^*$, maka dilanjutkan ke **Langkah 3**

Else

$\overline{X_{k+1}} = \overline{X_k} \setminus \{Q_{k,q_k}\}$ .....(2.18)

If $k+1 = D-d$, maka satu *leaf* telah dicapai dan dilanjutkan ke **Langkah 5**

Else

Menuju ke *level* yang berhubungan dan

$k = k+1$ .....(2.19)

kemudian kenbali ke **Langkah 1.**

Langkah 3

Node turunan dihubungkan dengan $Q_{k,qk}$ – edge (dan *subtree*-nya) kemungkinan di *cut-off*.

Fitur $Q_{k,qk}$ dikembalikan pada himpunan fitur yang tersedia pada pembuatan *tree*, yaitu :

$\Psi = \Psi \cup \{Q_{k,q_k}\}$ .....(2.20)

dan

$r = r + 1$ .....(2.21),

$Q_k = Q_k \setminus \{Q_{k,q_k}\}$ .....(2.22)

dan

$q_k = q_k - 1$ .....(2.23),

kemudian dilanjutkan dengan node sebelah kirinya dan kembali ke **Langkah 2**

Langkah 4

Backtracking:

$k = k - 1$ .....(2.24)

If $k = -1$, maka semua *tree* telah dicari sampai selesai dan algoritma selesai.

Else

fitur Q_{k,q_k} dikembalikan pada himpunan kandidat,

sehingga

$\overline{X_k} = \overline{X_{k+1}} \cup \{Q_{k,q_k}\}$ .....(2.25)

dan dilanjutkan ke **Langkah 3.**

Langkah 5

nilai *bound* di-*Update*:

$x^* = J_{k,q_k}$ .....(2.26)

himpunan bagian terbaik saat itu disimpan dalam

$X = \overline{X_{k+1}}$ .....(2.27)

dan dilanjutkan ke **Langkah 2**

2.3.1 Contoh Algoritma *Improved Branch & Bound*

Sebuah himpunan fitur awal dengan jumlah fiturnya sebanyak 5 fitur akan dipilih 2 fitur saja dengan fungsi kriterianya adalah $J(\overline{x}) = \sum_{\xi_i \in \overline{x}} i$.

Berdasarkan pada permasalahan diatas, maka penyelesaiannya adalah sebagai berikut:

Inisialisasi awal:

$D = 5$, $d = 2$, $J(\overline{x}) = \sum_{\xi_i \in \overline{x}} i$, $k = 0$, $\overline{x_0} = \{\xi_j | j = 1,2,3,4,5\} \rightarrow \overline{x_0} = \{\xi_1, \xi_2, \xi_3, \xi_4, \xi_5\}$,

$x = \{x_j | j = 1,2\} \rightarrow x = \{x_1, x_2\}$, $Y = \{\xi_1, \xi_2, \xi_3, \xi_4, \xi_5\}$, $\overline{x_0} = Y$, $\Psi = Y$, $r = D$,

$x^* = 1 + 2 = 3$

Langkah 1

Telah diketahui :

$k = 0$, $r = 5$

maka:

$q_k = r - (D - d - k - 1)$

$q_0 = 3$

Selama j = 1 sampai r → j = 1,2,3,4,5

$\psi_j \in \Psi$, dengan syarat : $J(\overline{x_0} \setminus \{\psi_{j_1}\}) \leq J(\overline{x_0} \setminus \{\psi_{j_2}\}) \leq \ldots \leq J(\overline{x_0} \setminus \{\psi_{j_5}\})$

untuk j = 1, didapatkan:

$\psi_1 = \xi_1$ sehingga $J(\overline{x_0} \setminus \{\xi_1\}) = J(2,3,4,5) = 14$

untuk j = 2, didapatkan:

$\psi_1 = \xi_2$ sehingga $J(\overline{x_0} \setminus \{\xi_2\}) = J(1,3,4,5) = 13$

untuk j = 3, didapatkan:

$\psi_1 = \xi_3$ sehingga $J(\overline{x_0} \setminus \{\xi_3\}) = J(1,2,4,5) = 12$

untuk j = 4, didapatkan:

$\psi_1 = \xi_4$ sehingga $J(\overline{x_0} \setminus \{\xi_4\}) = J(1,2,3,5) = 11$

untuk j = 5, didapatkan:

$\psi_1 = \xi_5$ sehingga $J(\overline{x_0} \setminus \{\xi_5\}) = J(1,2,3,4) = 10$

Karena harus sesuai syarat yang diminta, maka $\psi_1 = \xi_5, \psi_2 = \xi_4, \psi_3 = \xi_3, \psi_4 = \xi_2, \psi_5 = \xi_1$.

Kemudian dapat dipilih 3 fitur dari 5 fitur yang ada pada Ψ, yaitu:

$Q_{0,i} = \psi_{ji}$ untuk i = 1,...,q_0 → i = 1,2,3, sehingga:

$Q_{0,1} = \psi_{j_1} = \xi_5$

$Q_{0,2} = \psi_{j_2} = \xi_4$

$Q_{0,3} = \psi_{j_3} = \xi_3$

dan $Q_0 = \{\xi_5, \xi_4, \xi_3\}$

dan untuk $J_{0,i} = J(\overline{x_0} \setminus \{\psi_{j_i}\})$ untuk i = 1,...,q_0 → i = 1,2,3, sehingga:

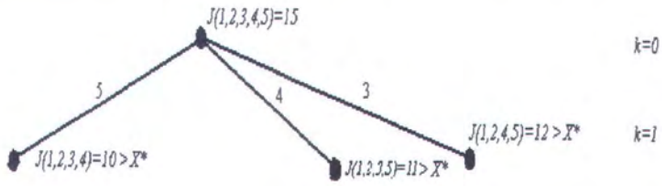
$J_{0,1} = J(\overline{x_0} \setminus \{\xi_5\}) = 10$

$J_{0,2} = J(\overline{x_0} \setminus \{\xi_4\}) = 11$

$J_{0,3} = J(\overline{x_0} \setminus \{\xi_3\}) = 12$

dan $J_0 = [10,11,12]^T$.

Dari perhitungan diatas didapat *node-node* keturunan dari *root* sebanyak 3 dan *tree*-nya dapat dilihat pada Gambar 2.23 berikut.



Gambar 2.23 Turunan *node* pada node *tree level* 0

Untuk menghindari evaluasi ganda, maka fitur ψ_{j_i} dihilangkan dari himpunan fitur Ψ, yaitu sebagai berikut:

$\Psi = \Psi \setminus Q_0$ dan $r = r - q_0$

$\Psi = \{\xi_1, \xi_2\}$ $r = 5 - 3 = 2$

Kemudian dilanjutkan ke **Langkah 2**.

Pada langkah ini terjadi perhitungan nilai kriteria sebanyak 5 kali, sehingga jumlah evaluasi kriterianya menjadi 5.

Langkah 2

Telah diketahui :

$k = 0,\ q_0 = 3,\ r = 2,\ \Psi = \{\xi_1, \xi_2\},\ \overline{x_0} = \{\xi_1, \xi_2, \xi_3, \xi_4, \xi_5\},\ x^* = 3$

Selanjutnya diperiksa:

- IF $q_0 = 0$

  Telah diketahui bahwa $q_0 = 3$, sehingga hal ini tidak memenuhi kondisi $q_0 = 0$.

- IF $J_{0,3} < x^*$

Dari langkah sebelumnya telah diketahui bahwa $J_{0,3} = 12$ dan $x^* = 3$, sehingga tidak memenuhi kondisi yang diminta.

- Else

$$\overline{x_{0+1}} = \overline{x_0} \setminus \{Q_{0,3}\}$$

$$\overline{x_1} = \{\xi_1, \xi_2, \xi_4, \xi_5\}$$

  - If $k+1 = D-d$

    Karena $k+1 = 1$ dan $D-d = 3$, maka hal ini tidak memenuhi kondisi yang diminta.

  - Else

$$k = k+1$$

$$k = 0+1$$

$$k = 1$$

kemudian dilanjutkan ke **Langkah 1**.

Langkah 1

Telah diketahui

$$k = 1,\ r = 2,\ \overline{x_1} = \{\xi_1, \xi_2, \xi_4, \xi_5\},\ \Psi = \{\xi_1, \xi_2\},\ x^* = 3$$

maka:

$$q_1 = 2 - (5 - 2 - 1 - 1)$$

$$q_1 = 1$$

Selama j = 1 sampai r → j = 1,2

$\psi_j \in \Psi$, dengan syarat : $J(\overline{x_1} \setminus \{\omega_{j_1}\}) \leq J(\overline{x_1} \setminus \{\omega_{j_2}\})$

untuk j = 1, didapatkan:

$\psi_1 = \xi_2$ sehingga $J(\overline{x_1} \setminus \{\xi_2\}) = J(1,4,5) = 10$

untuk j = 2, didapatkan:

$\psi_1 = \xi_1$ sehingga $J(\overline{x_1} \setminus \{\xi_1\}) = J(2,4,5) = 11$

Kemudian dapat dipilih 1 fitur dari 2 fitur yang ada pada Ψ, yaitu:

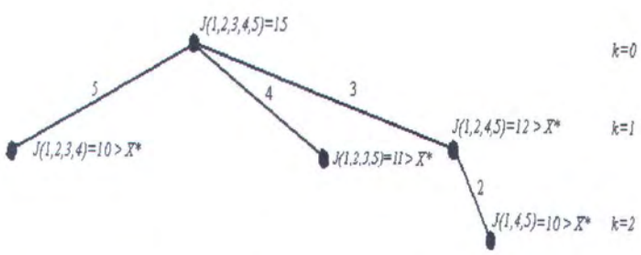
$Q_{1,i} = \psi_{J_i}$ untuk i = 1,...,q_1 → i = 1, sehingga:

$Q_{1,1} = \psi_1 = \xi_2$ dan $Q_1 = \{\xi_2\}$

dan untuk $J_{1,i} = (\overline{x_1} \setminus \{\psi_{J_i}\})$ untuk i = 1,...,q_1 → i = 1, sehingga:

$J_{1,1} = J(\overline{x_1} \setminus \{\xi_2\}) = 10$ dan $J_1 = [10]^T$.

Dari perhitungan diatas didapat *node-node* keturunan dari *node* paling kanan sebanyak 1 *node* dan *tree*-nya dapat dilihat pada Gambar 2.24 berikut.



Gambar 2.24 Turunan node pada *node tree level* 1

Untuk menghindari evaluasi ganda, maka fitur ψ_{J_i} dihilangkan dari himpunan fitur Ψ, yaitu sebagai berikut:

$\Psi = \Psi \setminus Q_1$ dan $r = r - q_1$

$\Psi = \{\xi_1\}$ $r = 2 - 1 = 1$

Kemudian dilanjutkan ke **Langkah 2**.

Pada langkah ini terjadi perhitungan nilai kriteria sebanyak 2 kali, sehingga jumlah evaluasi kriterianya menjadi 7.

Langkah 2

Telah diketahui :

$$k=1,\ q_1=1,\ r=1,\ \Psi=\{\xi_1\},\ \overline{x_1}=\{\xi_1,\xi_2,\xi_4,\xi_5\},\ x^*=3$$

Selanjutnya diperiksa:

- IF $q_1=0$

  Telah diketahui bahwa $q_1=1$, sehingga hal ini tidak memenuhi kondisi $q_1=0$.

- IF $J_{1,1}<x^*$

  Dari langkah sebelumnya telah diketahui bahwa $J_{1,1}=10$ dan $x^*=3$, sehingga tidak memenuhi kondisi yang diminta.

- Else

  $$\overline{x_{1+1}}=\overline{x_1}\setminus\{Q_{1,1}\}$$

  $$\overline{x_2}=\{\xi_1,\xi_4,\xi_5\}$$

  - If $k+1=D-d$

    Karena $k+1=2$ dan $D-d=3$, maka hal ini tidak memenuhi kondisi yang diminta.

  - Else

    $$k=k+1$$

    $$k=1+1$$

    $$k=2$$

Kemudian dilanjutkan ke **Langkah 1**.

Langkah 1

Telah diketahui

$k = 2$, $r = 1$, $\overline{x_2} = \{\xi_1, \xi_4, \xi_5\}$, $\Psi = \{\xi_1\}$, $x^* = 3$

maka:

$$q_2 = 1 - (5 - 2 - 2 - 1)$$

$$q_2 = 1$$

Selama j = 1 sampai r → j = 1

$\psi_j \in \Psi$,

untuk j = 1, didapatkan:

$\psi_1 = \xi_1$ sehingga $J(\overline{x_2} \setminus \{\xi_1\}) = J(4,5) = 9$

Kemudian dapat dipilih 1 fitur dari 1 fitur yang ada pada Ψ, yaitu:

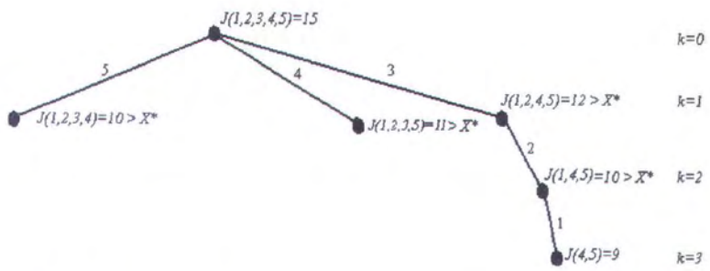
$Q_{2,i} = \psi_{j_i}$ untuk i = 1,...,q_2 → i = 1, sehingga:

$Q_{2,1} = \psi_1 = \xi_1$ dan $Q_2 = \{\xi_1\}$

dan untuk $J_{2,i} = (\overline{x_2} \setminus \{\psi_{J_i}\})$ untuk i = 1,...,q_2 → i = 1, sehingga:

$J_{2,1} = J(\overline{x_2} \setminus \{\xi_1\}) = 9$ dan $J_2 = [9]^T$.

Dari perhitungan diatas didapat *node-node* keturunan dari *node* paling kanan sebanyak 1 *node* dan *tree*-nya dapat dilihat pada Gambar 2.25 berikut.



Gambar 2.25 Turunan node pada node *tree* level 2

Untuk menghindari evaluasi ganda, maka fitur ψ_{j_i} dihilangkan dari himpunan fitur Ψ, yaitu sebagai berikut:

$\Psi = \Psi \setminus Q_2$ dan $r = r - q_2$

$\Psi = \phi$ $r = 1 - 1 = 0$

Kemudian dilanjutkan ke **Langkah 2**.

Pada langkah ini terjadi perhitungan nilai kriteria sebanyak 1 kali, sehingga jumlah evaluasi kriterianya menjadi 8.

Langkah 2

Telah diketahui :

$k = 2$, $q_2 = 1$, $r = 0$, $\Psi = \phi$, $\overline{x_2} = \{\xi_1, \xi_4, \xi_5\}$, $x^* = 3$

Selanjutnya diperiksa:

- IF $q_2 = 0$

  Telah diketahui bahwa $q_2 = 1$, sehingga hal ini tidak memenuhi kondisi $q_2 = 0$.

- IF $J_{2,1} < x^*$

  Dari langkah sebelumnya telah diketahui bahwa $J_{2,1} = 9$ dan $x^* = 3$, sehingga tidak memenuhi kondisi yang diminta.

- Else

$$\overline{x_{2+1}} = \overline{x_2} \setminus \{Q_{2,1}\}$$

$$\overline{x_3} = \{\xi_4, \xi_5\}$$

  - If $k+1 = D-d$

    Karena $k+1=3$ dan $D-d=3$, maka memenuhi kondisi yang diminta.

    Sehingga *leaf* ditemukan

Kemudian dilanjutkan ke **Langkah 5**.

Langkah 5

Nilai *bound* diupdate sebagai berikut:

$$x^* = J_{2,1}$$

$$x^* = 9$$

dan

$$x = \overline{x_{2+1}}$$

$$x = \overline{x_3}$$

$$x = \{\xi_4, \xi_5\}$$

Kemudian dilanjutkan ke **Langkah 2.**

Langkah 2

Telah diketahui :

$$k=2,\ q_2=1,\ r=0,\ \Psi=\phi,\ \overline{x_2}=\{\xi_1, \xi_4, \xi_5\},\ x^*=9$$

Selanjutnya diperiksa:

- IF $q_2 = 0$

Telah diketahui bahwa $q_2 = 1$, sehingga hal ini tidak memenuhi kondisi $q_2 = 0$.

- IF $J_{2,1} < x^*$

  Karena $J_{2,1} = 9$ dan $x^* = 9$, sehingga $J_{2,1} = x^*$. Kemudian dilanjutkan ke **Langkah 3**.

Langkah 3

Mengembalikan fitur $Q_{2,1}$ pada himpunan fitur Ψ, yaitu:

- $\Psi = \Psi \cup \{Q_{2,1}\}$ dan $r = r + 1$

  $\Psi = \{\xi_1\}$ $r = 0 + 1 = 1$

- $Q_2 = Q_2 \setminus \{Q_{2,1}\}$ dan $q_2 = q_2 - 1$

  $Q_2 = \phi$ $q_2 = 1 - 1 = 0$.

Kemudian dilanjutkan ke **Langkah 2.**

Langkah 2

Telah diketahui :

$k = 2$, $q_2 = 0$, $r = 1$, $\Psi = \{\xi_1\}$, $\overline{x_2} = \{\xi_1, \xi_4, \xi_5\}$, $x^* = 9$

Selanjutnya diperiksa:

- IF $q_2 = 0$

  Karena $q_2 = 0$, sehingga memenuhi kondisi $q_2 = 0$, maka dilanjutkan ke **Langkah 4**.

Langkah 4

Backtracking:

- $k = k - 1$

$k = 2 - 1$

$k = 1$

- If $k = -1$

  Karena $k = 1$, maka hal ini tidak memenuhi kondisi yang diminta.

- Else

  Mengembalikan fitur $Q_{1,1}$ pada himpunan kandidat, sehingga :

  $\overline{X_1} = \overline{X_2} \cup \{Q_{1,1}\}$

  $\overline{x_1} = \{\xi_1, \xi_2, \xi_4, \xi_5\}$

  Kemudian dilanjutkan ke **Langkah 3.**

Langkah 3

Mengembalikan fitur $Q_{1,1}$ pada himpunan fitur Ψ, yaitu:

- $\Psi = \Psi \cup \{Q_{1,1}\}$ dan $r = r + 1$

  $\Psi = \{\xi_1, \xi_2\}$ $r = 1 + 1 = 2$

- $Q_1 = Q_1 \setminus \{Q_{1,1}\}$ dan $q_1 = q_1 - 1$

  $Q_1 = \phi$ $q_1 = 1 - 1 = 0$.

Kemudian dilanjutkan ke **Langkah 2.**

Langkah 2

Telah diketahui :

$k = 1,\ q_1 = 0,\ r = 2,\ \Psi = \{\xi_1, \xi_2\},\ \overline{x_1} = \{\xi_1, \xi_2, \xi_4, \xi_5\},\ x^* = 9$

Selanjutnya diperiksa:

- IF $q_1 = 0$

  Karena $q_1 = 0$, sehingga memenuhi kondisi $q_1 = 0$, maka dilanjutkan ke **Langkah 4**.

Langkah 4

Backtracking:

- $k = k - 1$

  $k = 1 - 1$

  $k = 0$

  - If $k = -1$

    Karena $k = 0$, maka hal ini tidak memenuhi kondisi yang diminta.

  - Else

    Mengembalikan fitur $Q_{0,3}$ pada himpunan kandidat, sehingga :

    $\overline{X_0} = \overline{X_1} \cup \{Q_{0,3}\}$

    $\overline{x_0} = \{\xi_1, \xi_2, \xi_3, \xi_4, \xi_5\}$

    Kemudian dilanjutkan ke **Langkah 3.**

Langkah 3

Mengembalikan fitur $Q_{0,3}$ pada himpunan fitur Ψ, yaitu:

- $\Psi = \Psi \cup \{Q_{0,3}\}$ dan $r = r + 1$

  $\Psi = \{\xi_1, \xi_2, \xi_3\}$ $r = 2 + 1 = 3$

- $Q_0 = Q_0 \setminus \{Q_{0,3}\}$ dan $q_0 = q_0 - 1$

$Q_1 = \{\xi_5, \xi_4\}$ $q_0 = 3 - 1 = 2$.

Kemudian dilanjutkan ke **Langkah 2.**

Langkah 2

Telah diketahui :

$k = 0$, $q_0 = 2$, $r = 3$, $\Psi = \{\xi_1, \xi_2, \xi_3\}$, $\overline{x_0} = \{\xi_1, \xi_2, \xi_3, \xi_4, \xi_5\}$, $x^* = 9$

Selanjutnya diperiksa:

- IF $q_0 = 0$

  Telah diketahui bahwa $q_0 = 2$, sehingga hal ini tidak memenuhi kondisi $q_0 = 0$.

- IF $J_{0,2} < x^*$

  Dari langkah sebelumnya telah diketahui bahwa $J_{0,2} = 11$ dan $x^* = 9$, sehingga tidak memenuhi kondisi yang diminta.

- Else

  $\overline{x_{0+1}} = \overline{x_0} \setminus \{Q_{0,2}\}$

  $\overline{x_1} = \{\xi_1, \xi_2, \xi_3, \xi_5\}$

  - If $k + 1 = D - d$

    Karena $k + 1 = 1$ dan $D - d = 3$, maka hal ini tidak memenuhi kondisi yang diminta.

  - Else

    $k = k + 1$

    $k = 0 + 1$

    $k = 1$

Kemudian dilanjutkan ke **Langkah 1**.

Langkah 1

Telah diketahui

$k = 1$, $r = 3$, $\overline{x_1} = \{\xi_1, \xi_2, \xi_3, \xi_5\}$, $\Psi = \{\xi_1, \xi_2, \xi_3\}$, $x^* = 9$

maka:

$q_1 = 3 - (5 - 2 - 1 - 1)$

$q_1 = 2$

Selama j = 1 sampai r → j = 1,2,3

$\psi_j \in \Psi$, dengan syarat : $J(\overline{x_1} \setminus \{\psi_{j_1}\}) \le J(\overline{x_1} \setminus \{\psi_{j_2}\}) \le J(\overline{x_1} \setminus \{\psi_{j_3}\})$

untuk j = 1, didapatkan:

$\psi_1 = \xi_3$ sehingga $J(\overline{x_1} \setminus \{\xi_3\}) = J(1,2,5) = 8$

untuk j = 2, didapatkan:

$\psi_2 = \xi_2$ sehingga $J(\overline{x_1} \setminus \{\xi_2\}) = J(1,3,5) = 9$

untuk j = 3, didapatkan:

$\psi_3 = \xi_1$ sehingga $J(\overline{x_1} \setminus \{\xi_1\}) = J(2,3,5) = 10$

Kemudian dapat dipilih 2 fitur dari 3 fitur yang ada pada Ψ, yaitu:

$Q_{1,i} = \psi_{j_i}$ untuk i = 1,...,q_1 → i = 1,2, sehingga:

$Q_{1,1} = \psi_1 = \xi_3$

$Q_{1,2} = \psi_2 = \xi_2$

dan

$Q_1 = \{\xi_3, \xi_2\}$

dan untuk $J_{1,i} = (\overline{x_1} \setminus \{\psi_{j_i}\})$ untuk i = 1,...,q_1 → i = 1,2, sehingga:

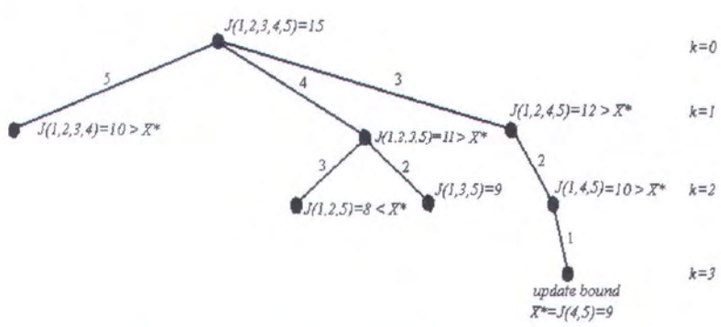
$$J_{1,1} = J(\overline{x_1} \setminus \{\xi_3\}) = 8$$

$$J_{1,2} = J(\overline{x_1} \setminus \{\xi_2\}) = 9$$

dan

$$J_1 = [8,9]^T .$$

Dari perhitungan diatas didapat *node-node* keturunan dari *node* pada level 1 sebanyak 2 *node* dan *tree*-nya dapat dilihat pada Gambar 2.26 berikut.



Gambar 2.26 Turunan *node* pada node *tree level* 1

Untuk menghindari evaluasi ganda, maka fitur ψ_{j_i} dihilangkan dari himpunan fitur Ψ, yaitu sebagai berikut:

$\Psi = \Psi \setminus Q_1$ dan $r = r - q_1$

$\Psi = \{\xi_1\}$ $r = 3 - 2 = 1$

Kemudian dilanjutkan ke **Langkah 2**.

Pada langkah ini terjadi perhitungan nilai kriteria sebanyak 3 kali, sehingga jumlah evaluasi kriterianya menjadi 11.

Langkah 2

Telah diketahui :

$$k = 1,\ q_1 = 2,\ r = 1,\ \Psi = \{\xi_1\},\ \overline{x_1} = \{\xi_1, \xi_2, \xi_3, \xi_5\},\ x^* = 9$$

Selanjutnya diperiksa:

- IF $q_1 = 0$

  Telah diketahui bahwa $q_1 = 2$, sehingga hal ini tidak memenuhi kondisi $q_1 = 0$.

- IF $J_{1,2} < x^*$

  Karena $J_{1,2} = 9$ dan $x^* = 9$, sehingga $J_{1,2} = x^*$. Kemudian dilanjutkan ke **Langkah 3**.

Langkah 3

Mengembalikan fitur $Q_{1,2}$ pada himpunan fitur Ψ, yaitu:

- $\Psi = \Psi \cup \{Q_{1,2}\}$ dan $r = r + 1$

  $\Psi = \{\xi_1, \xi_2\}$ $r = 1 + 1 = 2$

- $Q_1 = Q_1 \setminus \{Q_{1,2}\}$ dan $q_1 = q_1 - 1$

  $Q_1 = \{\xi_3\}$ $q_1 = 2 - 1 = 1$.

Kemudian dilanjutkan ke **Langkah 2.**

Langkah 2

Telah diketahui :

$k = 1$, $q_1 = 1$, $r = 2$, $\Psi = \{\xi_1, \xi_2\}$, $\overline{x_1} = \{\xi_1, \xi_2, \xi_3, \xi_5\}$, $x^* = 9$

Selanjutnya diperiksa:

- IF $q_1 = 0$

  Telah diketahui bahwa $q_1 = 1$, sehingga hal ini tidak memenuhi kondisi $q_1 = 0$.

- IF $J_{1,1} < x^*$

Karena $J_{1,1} = 8$ dan $x^* = 9$, sehingga memenuhi kondisi $J_{1,1} < x^*$.

Kemudian dilanjutkan ke **Langkah 3**.

Langkah 3

Mengembalikan fitur $Q_{1,2}$ pada himpunan fitur Ψ, yaitu:

- $\Psi = \Psi \cup \{Q_{1,1}\}$ dan $r = r + 1$

  $\Psi = \{\xi_1, \xi_2, \xi_3\}$ $r = 2 + 1 = 3$

- $Q_1 = Q_1 \setminus \{Q_{1,1}\}$ dan $q_1 = q_1 - 1$

  $Q_1 = \phi$ $q_1 = 1 - 1 = 0$.

Kemudian dilanjutkan ke **Langkah 2.**

Langkah 2

Telah diketahui :

$k = 1$, $q_1 = 0$, $r = 3$, $\Psi = \{\xi_1, \xi_2, \xi_3\}$, $\overline{x_1} = \{\xi_1, \xi_2, \xi_3, \xi_5\}$, $x^* = 9$

Selanjutnya diperiksa:

- IF $q_1 = 0$

  Telah diketahui bahwa $q_1 = 0$. Sehingga memenuhi kondisi $q_1 = 0$. Kemudian dilanjutkan ke **Langkah 4**.

Langkah 4

Backtracking:

- $k = k - 1$

  $k = 1 - 1$

  $k = 0$

  - If $k = -1$

Karena $k = 0$, maka hal ini tidak memenuhi kondisi yang diminta.

- Else

  Mengembalikan fitur $Q_{0,2}$ pada himpunan kandidat, sehingga :

  $\overline{X_0} = \overline{X_1} \cup \{Q_{0,2}\}$

  $\overline{x_0} = \{\xi_1, \xi_2, \xi_3, \xi_4, \xi_5\}$

  Kemudian dilanjutkan ke **Langkah 3.**

Langkah 3

Mengembalikan fitur $Q_{0,2}$ pada himpunan fitur Ψ, yaitu:

- $\Psi = \Psi \cup \{Q_{0,2}\}$ dan $r = r + 1$

  $\Psi = \{\xi_1, \xi_2, \xi_3, \xi_4\}$ $r = 3 + 1 = 4$

- $Q_0 = Q_0 \setminus \{Q_{0,2}\}$ dan $q_0 = q_0 - 1$

  $Q_1 = \{\xi_5\}$ $q_0 = 2 - 1 = 1$.

Kemudian dilanjutkan ke **Langkah 2.**

Langkah 2

Telah diketahui :

$k = 0$, $q_0 = 1$, $r = 4$, $\Psi = \{\xi_1, \xi_2, \xi_3, \xi_4\}$, $\overline{x_0} = \{\xi_1, \xi_2, \xi_3, \xi_4, \xi_5\}$, $x^* = 9$

Selanjutnya diperiksa:

- IF $q_0 = 0$

  Telah diketahui bahwa $q_0 = 1$. Sehingga hal ini tidak memenuhi kondisi $q_0 = 0$.

- IF $J_{0,1} < x^*$

  Dari langkah sebelumnya telah diketahui bahwa $J_{0,2} = 10$ dan $x^* = 9$, sehingga tidak memenuhi kondisi yang diminta.

- Else

  $$\overline{x_{0+1}} = \overline{x_0} \setminus \{Q_{0,1}\}$$

  $$\overline{x_1} = \{\xi_1, \xi_2, \xi_3, \xi_4\}$$

  - If $k+1 = D-d$

    Karena $k+1 = 1$ dan $D-d = 3$, maka hal ini tidak memenuhi kondisi yang diminta.

  - Else

    $$k = k+1$$
    $$k = 0+1$$
    $$k = 1$$

  Kemudian dilanjutkan ke **Langkah 1**.

Langkah 1

Telah diketahui

$k = 1$, $r = 4$, $\overline{x_1} = \{\xi_1, \xi_2, \xi_3, \xi_4\}$, $\Psi = \{\xi_1, \xi_2, \xi_3, \xi_4\}$, $x^* = 9$

maka:

$$q_1 = 4 - (5 - 2 - 1 - 1)$$

$$q_1 = 3$$

Selama j = 1 sampai r → j = 1,2,3,4

$\psi_j \in \Psi$, dengan syarat : $J(\overline{x_1} \setminus \{\psi_{j_3}\}) \le J(\overline{x_1} \setminus \{\psi_{j_2}\}) \le J(\overline{x_1} \setminus \{\psi_{j_5}\}) \le J(\overline{x_1} \setminus \{\psi_{j_4}\})$

untuk j = 1, didapatkan:

$\psi_1 = \xi_4$ sehingga $J(\overline{x_1} \setminus \{\xi_4\}) = J(1,2,3) = 6$

untuk j = 2, didapatkan:

$\psi_2 = \xi_3$ sehingga $J(\overline{x_1} \setminus \{\xi_3\}) = J(1,2,4) = 7$

untuk j = 3, didapatkan:

$\psi_3 = \xi_2$ sehingga $J(\overline{x_1} \setminus \{\xi_2\}) = J(1,3,4) = 8$

untuk j = 4, didapatkan:

$\psi_4 = \xi_1$ sehingga $J(\overline{x_1} \setminus \{\xi_1\}) = J(2,3,4) = 9$

Kemudian dapat dipilih 3 fitur dari 4 fitur yang ada pada Ψ, yaitu:

$Q_{1,i} = \psi_{J_1}$ untuk i = 1,...,q_1 → i = 1,2,3, sehingga:

$Q_{1,1} = \psi_1 = \xi_4$

$Q_{1,2} = \psi_2 = \xi_3$

$Q_{1,2} = \psi_2 = \xi_2$

dan

$Q_1 = \{\xi_4, \xi_3, \xi_2\}$

dan untuk $J_{1,i} = (\overline{x_1} \setminus \{\psi_{J_i}\})$ untuk i = 1,...,q_1 → i = 1,2,3, sehingga:

$J_{1,1} = J(\overline{x_1} \setminus \{\xi_4\}) = 6$

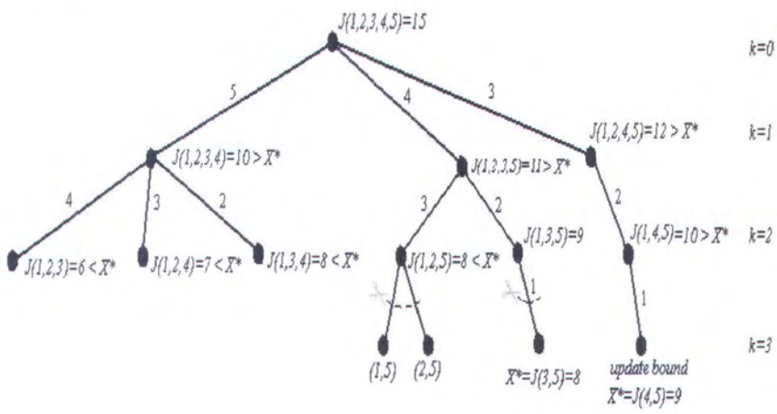
$J_{1,2} = J(\overline{x_1} \setminus \{\xi_3\}) = 7$

$J_{1,3} = J(\overline{x_1} \setminus \{\xi_2\}) = 8$

dan

$J_1 = [6,7,8]^T$.

Dari perhitungan diatas didapat node-node keturunan dari *node* pada level 1 sebanyak 3 *node* dan *tree*-nya dapat dilihat pada Gambar 2.27 berikut.



Gambar 2.27 Turunan node pada *node tree level* 1

Untuk menghindari evaluasi ganda, maka fitur ψ_{j_i} dihilangkan dari himpunan fitur Ψ, yaitu sebagai berikut:

- $\Psi = \Psi \setminus Q_1$ dan $r = r - q_1$

  $\Psi = \{\xi_1\}$ $r = 4 - 3 = 1$

Kemudian dilanjutkan ke **Langkah 2**.

Langkah 2

Telah diketahui :

$k = 1$, $q_1 = 3$, $r = 1$, $\Psi = \{\xi_1\}$, $\overline{x_1} = \{\xi_1, \xi_2, \xi_3, \xi_4\}$, $x^* = 9$

Selanjutnya diperiksa:

- IF $q_1 = 0$

  Telah diketahui bahwa $q_1 = 3$, sehingga hal ini tidak memenuhi kondisi $q_1 = 0$.

- IF $J_{1,3} < x^*$

  Karena $J_{1,3} = 8$ dan $x^* = 9$, sehingga memenuhi kondisi $J_{1,3} < x^*$.

  Kemudian dilanjutkan ke **Langkah 3**.

Langkah 3

Mengembalikan fitur $Q_{1,3}$ pada himpunan fitur Ψ, yaitu:

- $\Psi = \Psi \cup \{Q_{1,3}\}$ dan $r = r + 1$

  $\Psi = \{\xi_1, \xi_2\}$ $r = 1 + 1 = 2$

- $Q_1 = Q_1 \setminus \{Q_{1,3}\}$ dan $q_1 = q_1 - 1$

  $Q_1 = \{\xi_4, \xi_3\}$ $q_1 = 3 - 1 = 2$.

Kemudian dilanjutkan ke **Langkah 2.**

Langkah 2

Telah diketahui :

$k = 1$, $q_1 = 2$, $r = 2$, $\Psi = \{\xi_1, \xi_2\}$, $\overline{x_1} = \{\xi_1, \xi_2, \xi_3, \xi_4\}$, $x^* = 9$

Selanjutnya diperiksa:

- IF $q_1 = 0$

  Telah diketahui bahwa $q_1 = 2$, sehingga hal ini tidak memenuhi kondisi $q_1 = 0$.

- IF $J_{1,2} < x^*$

  Karena $J_{1,2} = 7$ dan $x^* = 9$, sehingga memenuhi kondisi $J_{1,2} < x^*$.

  Kemudian dilanjutkan ke **Langkah 3**.

Langkah 3

Mengembalikan fitur $Q_{1,2}$ pada himpunan fitur Ψ, yaitu:

- $\Psi = \Psi \cup \{Q_{1,2}\}$ dan $r = r + 1$

  $\Psi = \{\xi_1, \xi_2, \xi_3\}$ $r = 2 + 1 = 3$

- $Q_1 = Q_1 \setminus \{Q_{1,2}\}$ dan $q_1 = q_1 - 1$

  $Q_1 = \{\xi_4\}$ $q_1 = 2 - 1 = 1$.

Kemudian dilanjutkan ke **Langkah 2.**

Langkah 2

Telah diketahui :

$k = 1$, $q_1 = 1$, $r = 3$, $\Psi = \{\xi_1, \xi_2, \xi_3\}$, $\overline{x_1} = \{\xi_1, \xi_2, \xi_3, \xi_4\}$, $x^* = 9$

Selanjutnya diperiksa:

- IF $q_1 = 0$

  Telah diketahui bahwa $q_1 = 1$, sehingga hal ini tidak memenuhi kondisi $q_1 = 0$.

- IF $J_{1,1} < x^*$

  Karena $J_{1,1} = 6$ dan $x^* = 9$, sehingga memenuhi kondisi $J_{1,1} < x^*$.

  Kemudian dilanjutkan ke **Langkah 3**.

Langkah 3

Mengembalikan fitur $Q_{1,1}$ pada himpunan fitur Ψ, yaitu:

- $\Psi = \Psi \cup \{Q_{1,1}\}$ dan $r = r + 1$

  $\Psi = \{\xi_1, \xi_2, \xi_3, \xi_4\}$ $r = 3 + 1 = 4$

- $Q_1 = Q_1 \setminus \{Q_{1,1}\}$ dan $q_1 = q_1 - 1$

  $Q_1 = \phi$ $q_1 = 1 - 1 = 0$.

Kemudian dilanjutkan ke **Langkah 2.**

Langkah 2

Telah diketahui :

$k = 1$, $q_1 = 0$, $r = 4$, $\Psi = \{\xi_1, \xi_2, \xi_3, \xi_4\}$, $\overline{x_1} = \{\xi_1, \xi_2, \xi_3, \xi_4\}$, $x^* = 9$

Selanjutnya diperiksa:

- IF $q_1 = 0$

  Telah diketahui bahwa $q_1 = 0$, sehingga memenuhi kondisi $q_1 = 0$. Kemudian dilanjutkan ke **Langkah 4**.

Langkah 4

Backtracking:

- $k = k - 1$

  $k = 1 - 1$

  $k = 0$

  - If $k = -1$

    Karena $k = 0$, maka hal ini tidak memenuhi kondisi yang diminta.

  - Else

    Mengembalikan fitur $Q_{0,1}$ pada himpunan kandidat, sehingga :

    $\overline{X_0} = \overline{X_1} \cup \{Q_{0,1}\}$

    $\overline{x_0} = \{\xi_1, \xi_2, \xi_3, \xi_4, \xi_5\}$

    Kemudian dilanjutkan ke **Langkah 3.**

Langkah 3

Mengembalikan fitur $Q_{0,1}$ pada himpunan fitur Ψ, yaitu:

- $\Psi = \Psi \cup \{Q_{0,1}\}$ dan $r = r + 1$

  $\Psi = \{\xi_1, \xi_2, \xi_3, \xi_4, \xi_5\}$ $r = 4 + 1 = 5$

- $Q_0 = Q_0 \setminus \{Q_{0,1}\}$ dan $q_0 = q_0 - 1$

  $Q_0 = \phi$ $q_0 = 1 - 1 = 0$.

Kemudian dilanjutkan ke **Langkah 2.**

Langkah 2

Telah diketahui :

$k = 0$, $q_0 = 0$, $r = 5$, $\Psi = \{\xi_1, \xi_2, \xi_3, \xi_4, \xi_5\}$, $\overline{x_0} = \{\xi_1, \xi_2, \xi_3, \xi_4, \xi_5\}$, $x^* = 9$

Selanjutnya diperiksa:

- IF $q_0 = 0$

  Telah diketahui bahwa $q_0 = 0$, sehingga memenuhi kondisi $q_0 = 0$. Kemudian dilanjutkan ke **Langkah 4**.

Langkah 4

Backtracking:

- $k = k - 1$

  $k = 0 - 1$

  $k = -1$

  - If $k = -1$

    Karena $k = -1$, maka memenuhi kondisi yang diminta, sehingga algoritma telah selesai.

Hasil dari penelusuran *tree* pada algoritma IBB secara keseluruhan dapat dilihat pada Gambar 2.27 dan didapatkan nilai *bound* adalah 9 dan fitur yang dipilih adalah fitur 4 dan fitur 5. Jumlah evaluasi kriteria yang terjadi pada algoritma IBB ini sebanyak 14.

Perbandingan antara algoritma BBB dan algoritma IBB secara sederhana adalah sebagai berikut:

a. Penelusuran *tree* pada algoritma IBB lebih pendek daripada penelusuran *tree* pada algoritma BBB.
b. Perhitungan evaluasi kriteria pada algoritma BBB lebih banyak dilakukan daripada pada algoritma IBB.
c. Adanya pengurutan node secara *ascending* pada *level* yang sama dalam algoritma IBB dapat mengurangi jumlah perhitungan evaluasi kriteria.
d. Algoritma IBB lebih *representatif* daripada algoritma BBB.

BAB III

METODE SELEKSI FITUR MENGGUNAKAN ALGORITMA *BRANCH & BOUND PARTIAL PREDICTION* DAN ALGORITMA *FAST BRANCH & BOUND*

Pada bab ini dijelaskan algoritma *Branch & Bound Partial Prediction* (BBPP) dan algoritma *Fast Branch & Bound* (FBB) yang digunakan dalam seleksi fitur. Prinsip dasar kedua algoritma ini hampir sama dengan algoritma *Improved Branch & Bound* (IBB) yang telah dijelaskan pada BAB II. Perbedaan kedua algoritma ini dengan algoritma *Improved Branch & Bound* (IBB) adalah terletak pada prediksi fitur kandidat, yang mana pada algoritma *Improved Branch & Bound* (IBB) tidak ada. Kedua algoritma ini akan dijelaskan secara detil pada subbab selanjutnya.

3.1 BRANCH & BOUND PARTIAL PREDICTION

Algoritma *Branch & Bound Partial Prediction* (BBPP) ini bertujuan untuk mengurutkan *node-node* pada tiap *level tree* dengan jumlah evaluasi kriteria yang kecil. Tujuan algoritma BBPP ini hampir sama dengan algoritma IBB, hanya saja pada algoritma BBPP ini ada tambahan dengan menggunakan sedikit prediksi. Algoritma ini membentuk *level* pohon yang berhubungan dengan beberapa tahap. Pertama, nilai kriteria turunan tiap fitur yang ada saat itu diprediksi dengan cepat untuk pembentukan *tree*. Fitur-fitur tersebut kemudian diurutkan secara menurun berdasarkan pada prediksi nilai kriteria turunan. Nilai kriteria sebenarnya dihitung

hanya untuk jumlah fitur yang diminta (dimulai dari fitur dengan prediksi nilai kriteria turunan yang paling tinggi) dan *level* pohon yang berhubungan dibentuk. Pada tahap ini, total jumlah perhitungan kriteria sebenarnya pada tiap *level* tetap sebanding dengan algoritma *Branch & Bound* dasar tanpa adanya pengurutan *node*.

Mekanisme prediksi yang digunakan oleh algoritma BBPP ini menggunakan pengumpulan statistik dari nilai kriteria turunan yang disebabkan oleh penghilangan sebagian fitur selama proses pencarian. Satu fitur beda dihilangkan dalam tahap pembentukan pohon dan informasi prediksi dikumpulkan secara terpisah untuk tiap fitur. Algoritma BBPP ini menggunakan vektor kontribusi yang merupakan vektor dari kontribusi fitur pada nilai kriteria dan dinotasikan dengan

$$A = [A_1, A_2, ..., A_D]^T \quad \text{.....(3.1).}$$

Vektor ini menyimpan rata-rata nilai kriteria turunan yang disebabkan oleh penghilangan satu fitur dari himpunan bagian *current* kandidat untuk tiap fitur. Algoritma ini juga menggunakan vektor kounter yang menyimpan jumlah evaluasi dari nilai kriteria turunan yang dilakukan pada tiap fitur tunggal. Vektor ini dinotasikan dengan

$$S = [S_1, S_2, ..., S_D]^T \quad \text{.....(3.2).}$$

Notasi-notasi lain yang digunakan dalam algoritma BBPP ini adalah notasi-notasi yang digunakan dalam algoritma IBB. Notasi-notasi ini telah dijelaskan dalam BABII.

Ketika algoritma menghilangkan beberapa fitur y_i dari himpunan bagian *current* kandidat dan menghitung nilai kriteria sebenarnya yang berhubungan $J(\overline{x_k} \setminus \{y_i\})$ pada level pohon ke-k, maka informasi prediksi di-*update* sebagai berikut:

$$A_{y_i} = \frac{A_{y_i} . S_{y_i} + \left(J(\overline{x_k}) - J(\overline{x_k} \setminus \{y_i\})\right) 1}{S_{y_i} + 1} \quad \text{.....(3.3)}$$

dan

$$S_{y_i} = S_{y_i} + 1 \quad \text{.....(3.4)}$$

dengan A_{y_i} dan S_{y_i} inisialisasi awalnya diset 0 untuk semua $i = 1,...,D$

3.1.1 Algoritma *Branch & Bound Partial Prediction*

Algoritma ini mempunyai tahap-tahap yang sama dengan algoritma IBB. Pertama, algoritma ini menggunakan informasi prediksi untuk menentukan jumlah turunan dari *current* node. Kemudian informasi prediksi diupdate sesuai dengan rumus 3.3 dan 3.4.

Langkah-langkah dari algoritma BBPP adalah sebagai berikut [SPK04]:

Inisialisasi awal:

$k = 0$, $\overline{x_0} = Y$, $\Psi = Y$, $r = D$ dan x^* merupakan nilai paling kecil yang mungkin

Langkah 1 : Memilih turunan dari *current node* untuk membentuk *level* pohon yang berhubungan

- $q_k = r - (D - d - k - 1)$ .....(3.5)
- dibentuk himpunan terurut Q_k dan vektor J_k sebagai berikut:

semua fitur $\psi_j \in \Psi$ dengan $j = 1,...,r$ diurutkan secara menurun berdasarkan pada nilai A_{ψ_j} dengan $j = 1,...,r$, dan harus memenuhi syarat

$$A_{\psi_{j_1}} \geq A_{\psi_{j_2}} \geq ... \geq A_{\psi_{j_r}} \quad \text{.....(3.6)}$$

dan kemudian dipilih fitur sebanyak q_k fitur dari himpunan fitur yang ada pada Ψ, yaitu dengan cara:

$$Q_{k,i} = \psi_{j_i} \quad \text{.....(3.7)}$$

untuk i = 1,...,q_k

$$J_{k,i} = J\left(\overline{X_k} \setminus \{\psi_{j_i}\}\right) \quad \text{.....(3.8)}$$

untuk i = 1,...,q_k

Untuk menghindari evaluasi ganda, maka fitur ψ_{j_i} dihilangkan dari Ψ, yaitu :

$$\Psi = \Psi \setminus Q_k \quad \text{.....(3.9)}$$

dan

$$r = r - q_k \quad \text{.....(3.10)}$$

Untuk langkah 2, 3, 4 dan 5 sama dengan algoritma IBB, yaitu sebagai berikut:

Langkah 2

node turunan paling kanan diperiksa (dihubungkan dengan $Q_{k,qk}$ – edge) :

- If $q_k = 0$, semua turunan telah diuji dan langsung dilanjutkan ke **Langkah 4** (*backtracking*)
- If $J_{k,q_k} < x^*$, kemudian dilanjutkan ke **Lamgkah 3**
- Else

$\overline{X_{k+1}} = \overline{X_k} \setminus \{Q_{k,q_k}\}$ .....(3.11)

If $k+1 = D-d$, maka *leaf* telah ditemukan dan dilanjutkan ke **Langkah 5**

Else

Dilanjutkan ke *level* selanjutnya dengan

$k = k+1$ .....(3.12)

kemudian dilanjutkan ke **Langkah 1**

Langkah 3

Node turunan yang dihubungkan dengan Q_{k,q_k} *- edge* (dan *subtree*-nya) kemungkinan di *cut off*:

Mengembalikan fitur Q_{k,q_k} pada himpunan fitur yang tersedia pada pembuatan *tree*, yaitu :

$\Psi = \Psi \cup \{Q_{k,q_k}\}$ .....(3.13)

dan

$r = r+1$ .....(3.14),

$Q_k = Q_k \setminus \{Q_{k,q_k}\}$ .....(3.15)

dan

$q_k = q_k - 1$ .....(3.16).

kemudian dilanjutkan dengan *node* sebelah kirinya, lalu dilanjutkan ke **Langkah 2**

Langkah 4

Backtracking:

- $k = k - 1$ .....(3.17)
- *If* $k = -1$, maka semua *tree* telah dicari secara keseluruhan dan algoritma selesai.
- *Else*

  Mengembalikan fitur Q_{k,q_k} pada himpunan kandidat, sehingga

  $\overline{X_k} = \overline{X_{k+1}} \cup \{Q_{k,q_k}\}$ .....(3.18)

  dan dilanjutkan ke **Langkah 3.**

Langkah 5

nilai *bound* di-*update*:

- $x^* = J_{k,q_k}$ .....(3.19)
- fitur terbaik disimpan dalam

  $X = \overline{X_{k+1}}$ .....(3.20)

  dan dilanjutkan ke **Langkah 2**

3.1.2 Contoh Algoritma *Branch & Bound Partial Prediction*

Permasalahan yang diambil sama dengan permasalahan pada bab sebelumnya yaitu misalkan ada himpunan fitur awal dengan jumlah fiturnya sebanyak 5 fitur. Dari himpunan fitur awal ini akan dipilih 2 fitur saja dengan fungsi kriterianya adalah $J(\bar{x}) = \sum_{\xi_i \in x} i$.

Berdasarkan pada permasalahan diatas maka penyelesaiannya menggunakan algoritma BBPP adalah sebagai berikut:

Inisialisasi awal:

$D=5$, $d=2$, $J(\overline{x})=\sum_{\xi_i \in x} i$, $k=0$, $\overline{x_0}=\{\xi_j | j=1,2,3,4,5\} \rightarrow \overline{x_0}=\{\xi_1,\xi_2,\xi_3,\xi_4,\xi_5\}$,

$x=\{x_j | j=1,2\} \rightarrow x=\{x_1,x_2\}$, $Y=\{\xi_1,\xi_2,\xi_3,\xi_4,\xi_5\}$, $\overline{x_0}=Y$, $\Psi=Y$, $r=D$,

$x^*=1+2=3$, $J(\overline{x_0})=1+2+3+4+5=15$

Langkah 1

Telah diketahui :

$k=0$, $r=5$

maka:

- $q_k=r-(D-d-k-1)$

  $q_0=5-(5-2-0-1)$

  $q_0=3$

- Untuk j = 1 sampai r , sehingga nilai j = 1, 2, 3, 4, 5, maka

  $\psi_j \in \varpi$, dengan syarat : $A_{\psi_{j1}} \geq A_{\psi_{j2}} \geq A_{\psi_{j3}} \geq A_{\psi_{j4}} \geq A_{\psi_{j5}}$

  untuk j = 1, didapatkan:

  $\psi_1=\xi_1$ sehingga

  $$A_1=\frac{A_1.S_1+\left(J(\overline{x_0})-J(\overline{x_0}\setminus\{\xi_1\})\right)1}{S_1+1}$$

  $$A_1=\frac{0.0+(15-14)1}{0+1}$$

  $A_1=1$

  dan

$$S_1 = S_1 + 1$$

$$S_1 = 0 + 1$$

$$S_1 = 1$$

untuk j = 2, didapatkan:

$\psi_1 = \xi_2$ sehingga

$$A_2 = \frac{A_2 . S_2 + \left(J\left(\overline{x_0}\right) - J\left(\overline{x_0} \setminus \{\xi_2\}\right)\right) 1}{S_2 + 1}$$

$$A_2 = \frac{0.0 + (15 - 13).1}{0 + 1}$$

$$A_2 = 2$$

dan

$$S_2 = S_2 + 1$$

$$S_2 = 0 + 1$$

$$S_2 = 1$$

untuk j = 3, didapatkan:

$\psi_1 = \xi_3$ sehingga

$$A_3 = \frac{A_3 . S_3 + \left(J\left(\overline{x_0}\right) - J\left(\overline{x_0} \setminus \{\xi_3\}\right)\right) 1}{S_3 + 1}$$

$$A_3 = \frac{0.0 + (15 - 12).1}{0 + 1}$$

$$A_3 = 3$$

dan

$$S_3 = S_3 + 1$$

$$S_3 = 0 + 1$$

$$S_3 = 1$$

untuk j = 4, didapatkan:

$\psi_1 = \xi_4$ sehingga

$$A_4 = \frac{A_4.S_4 + \left(J\left(\overline{x_0}\right) - J\left(\overline{x_0} \setminus \{\xi_4\}\right)\right)1}{S_4 + 1}$$

$$A_4 = \frac{0.0 + (15-11).1}{0+1}$$

$$A_4 = 4$$

dan

$$S_4 = S_4 + 1$$

$$S_4 = 0 + 1$$

$$S_4 = 1$$

untuk j = 5, didapatkan:

$\psi_1 = \xi_5$ sehingga

$$A_5 = \frac{A_5.S_5 + \left(J\left(\overline{x_0}\right) - J\left(\overline{x_0} \setminus \{\xi_5\}\right)\right)1}{S_5 + 1}$$

$$A_5 = \frac{0.0 + (15-10).1}{0+1}$$

$$A_5 = 5$$

dan

$$S_5 = S_5 + 1$$

$$S_5 = 0 + 1$$

$$S_5 = 1$$

- Karena harus sesuai syarat yang diminta, maka.

$\psi_1 = \xi_5$ dan $A_5 = 5$, $S_5 = 1$

$\psi_2 = \xi_4$ dan $A_4 = 4$, $S_4 = 1$

$\psi_3 = \xi_3$ dan $A_3 = 3$, $S_3 = 1$

$\psi_4 = \xi_2$ dan $A_2 = 2$, $S_2 = 1$

$\psi_5 = \xi_1$ dan $A_1 = 1$, $S_1 = 1$

sehingga $A_{\psi_{J1}} \geq A_{\psi_{J2}} \geq A_{\psi_{J3}} \geq A_{\psi_{J4}} \geq A_{\psi_{J5}}$

- Kemudian dapat dipilih 3 fitur dari 5 fitur yang ada pada Ψ, yaitu:

  $Q_{0,i} = \psi_{ji}$ untuk i = 1,...,q_0 sehingga nilai i = 1, 2, 3 dan didapatkan

  $Q_{0,1} = \psi_{j_1} = \xi_5$

  $Q_{0,2} = \psi_{j_2} = \xi_4$

  $Q_{0,3} = \psi_{j_3} = \xi_3$

  sehingga $Q_0 = \{\xi_5, \xi_4, \xi_3\}$

  dan untuk $J_{0,i} = J(\overline{x_0} \setminus \{\psi_{j_i}\})$ untuk i = 1,...,q_0 sehingga nilai i = 1,2,3 dan didapatkan:

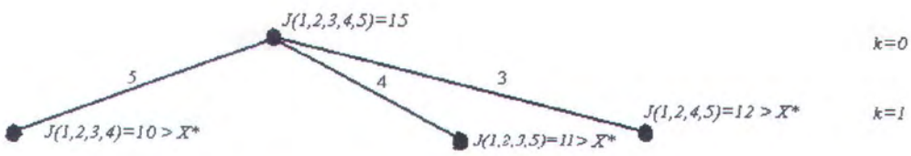
  $J_{0,1} = J(\overline{x_0} \setminus \{\xi_5\}) = 10$

  $J_{0,2} = J(\overline{x_0} \setminus \{\xi_4\}) = 11$

  $J_{0,3} = J(\overline{x_0} \setminus \{\xi_3\}) = 12$

  sehingga $J_0 = [10,11,12]^T$.

Dari perhitungan diatas didapat node-node keturunan dari *root* sebanyak 3 dan *tree*-nya dapat dilihat pada Gambar 3.1 berikut.



Gambar 3.1 Turunan node pada node *tree level* 0

Untuk menghindari duplikasi testing, maka fitur ψ_{j_i} dihilangkan dari himpunan fitur Ψ, yaitu sebagai berikut:

- $\Psi = \Psi \setminus Q_0$ dan $r = r - q_0$

  $\Psi = \{\xi_1, \xi_2\}$ $r = 5 - 3 = 2$

Kemudian dilanjutkan ke **Langkah 2**.

Pada langkah ini terjadi perhitungan nilai kriteria sebenarnya sebanyak 3 kali, sehingga jumlah evaluasi kriterianya menjadi 3.

Langkah 2

Telah diketahui :

$k = 0$, $q_0 = 3$, $r = 2$, $\Psi = \{\xi_1, \xi_2\}$, $\overline{x_0} = \{\xi_1, \xi_2, \xi_3, \xi_4, \xi_5\}$, $x^* = 3$

Selanjutnya diperiksa:

- *IF* $q_0 = 0$

  Telah diketahui bahwa $q_0 = 3$. Sehingga hal ini tidak memenuhi kondisi $q_0 = 0$.

- *IF* $J_{0,3} < x^*$

  Dari langkah sebelumnya telah diketahui bahwa $J_{0,3} = 12$ dan $x^* = 3$, sehingga tidak memenuhi kondisi yang diminta.

- *Else*

$\overline{x_{0+1}} = \overline{x_0} \setminus \{\xi_{0,3}\}$

$\overline{x_1} = \{\xi_1, \xi_2, \xi_4, \xi_5\}$

- If $k+1 = D-d$

  Karena $k+1=1$ dan $D-d=3$, maka hal ini tidak memenuhi kondisi yang diminta.

- Else

  $k = k+1$

  $k = 0+1$

  $k = 1$

Kemudian dilanjutkan ke **Langkah 1**.

Langkah 1

Telah diketahui

$k=1$, $r=2$, $\overline{x_1} = \{\xi_1, \xi_2, \xi_4, \xi_5\}$, $\Psi = \{\xi_1, \xi_2\}$, $x^* = 3$

maka:

- $q_1 = 2-(5-2-1-1)$

  $q_1 = 1$

Untuk j = 1 sampai r sehingga nilai j = 1,2

$\psi_j \in \Psi$, dengan syarat : $A_{\psi_{J1}} \geq A_{\psi_{J2}}$

untuk j = 1, didapatkan:

$\psi_1 = \xi_2$ sehingga

$$A_2 = \frac{A_2.S_2 + \left(J\left(\overline{x_1}\right) - J\left(\overline{x_1} \setminus \{\xi_2\}\right)\right)1}{S_2+1}$$

$$A_2 = \frac{2.1 + (12 - 10).1}{1 + 1}$$

$$A_2 = 2$$

dan

$$S_2 = S_2 + 1$$

$$S_2 = 1 + 1$$

$$S_2 = 2$$

untuk j = 2, didapatkan:

$\psi_2 = \xi_1$ sehingga

$$A_1 = \frac{A_1.S_1 + \left(J(\overline{x_1}) - J(\overline{x_1} \setminus \{\xi_1\})\right).1}{S_1 + 1}$$

$$A_1 = \frac{1.1 + (12 - 11).1}{1 + 1}$$

$$A_1 = 1$$

dan

$$S_1 = S_1 + 1$$

$$S_1 = 1 + 1$$

$$S_1 = 2$$

- Kemudian dapat dipilih 1 fitur dari 2 fitur yang ada pada Ψ, yaitu:

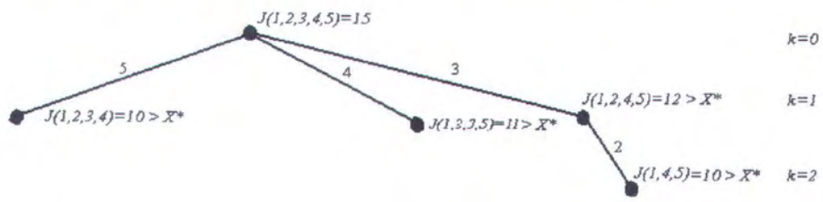
$Q_{1,i} = \psi_{J_i}$ untuk i = 1,...,q_1 sehingga nilai i = 1 dan didapatkan:

$Q_{1,1} = \psi_1 = \xi_2$ dan $Q_1 = \{\xi_2\}$

dan $J_{1,i} = (\overline{x_1} \setminus \{\psi_{J_i}\})$ untuk i = 1,...,q_1 sehingga nilai i = 1 dan didapatkan:

$J_{1,1} = J(\overline{x_1} \setminus \{\xi_2\}) = 10$ dan $J_1 = [10]^T$.

Dari perhitungan diatas didapat node-node keturunan dari *node* paling kanan sebanyak 1 *node* dan *tree*-nya dapat dilihat pada Gambar 3.2 berikut.



Gambar 3.2 Turunan node pada node *tree* level 1

Untuk menghindari duplikasi testing, maka fitur ψ_{j_i} dihilangkan dari himpunan fitur Ψ, yaitu sebagai berikut:

- $\Psi = \Psi \setminus Q_1$ dan $r = r - q_1$

  $\Psi = \{\xi_1\}$ $r = 2 - 1 = 1$

Kemudian dilanjutkan ke **Langkah 2**.

Pada langkah ini terjadi perhitungan nilai kriteria sebenarnya sebanyak 1 kali, sehingga jumlah evaluasi kriterianya menjadi 4.

Langkah 2

Telah diketahui :

$k = 1$, $q_1 = 1$, $r = 1$, $\Psi = \{\xi_1\}$, $\overline{x_1} = \{\xi_1, \xi_2, \xi_4, \xi_5\}$, $x^* = 3$

Selanjutnya diperiksa:

- IF $q_1 = 0$

  Telah diketahui bahwa $q_1 = 1$. Sehingga hal ini tidak memenuhi kondisi $q_1 = 0$.

- IF $J_{1,1} < x^*$

Dari langkah sebelumnya telah diketahui bahwa $J_{1,1}=10$ dan $x^*=3$, sehingga tidak memenuhi kondisi yang diminta.

- Else

  $\overline{x_{1+1}}=\overline{x_1}\setminus\{Q_{1,1}\}$

  $\overline{x_2}=\{\xi_1,\xi_4,\xi_5\}$

  - If $k+1=D-d$

    Karena $k+1=2$ dan $D-d=3$, maka hal ini tidak memenuhi kondisi yang diminta.

  - Else

    $k=k+1$

    $k=1+1$

    $k=2$

Kemudian dilanjutkan ke **Langkah 1**.

Langkah 1

Telah diketahui

$k=2$, $r=1$, $\overline{x_2}=\{\xi_1,\xi_4,\xi_5\}$, $\Psi=\{\xi_1\}$, $x^*=3$

maka:

- $q_2=1-(5-2-2-1)$

  $q_1=1$

- Untuk j = 1 sampai r sehingga nilai j = 1

  $\psi_j\in\Psi$,

  untuk j = 1, didapatkan:

$\psi_1 = \xi_1$ sehingga

$$A_1 = \frac{A_1 . S_1 + \left(J\left(\overline{x_2}\right) - J\left(\overline{x_2} \setminus \{\xi_1\}\right)\right).1}{S_1 + 1}$$

$$A_1 = \frac{1.2 + (10 - 9).1}{2 + 1}$$

$$A_1 = 1$$

dan

$$S_1 = S_1 + 1$$

$$S_1 = 2 + 1$$

$$S_1 = 3$$

- Kemudian dipilih 1 fitur dari 1 fitur yang ada pada Ψ, yaitu:

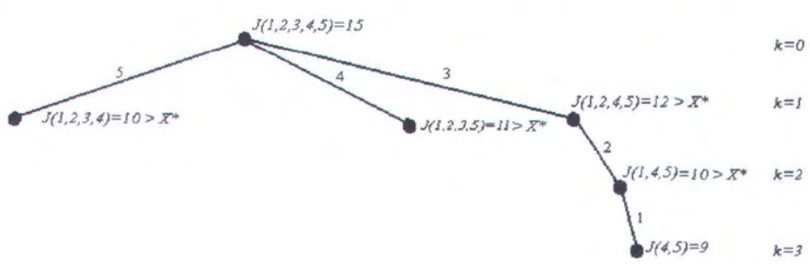
  $Q_{2,i} = \psi_{j_i}$ untuk i = 1,...,q_2 sehingga nilai i = 1 dan didapatkan:

  $Q_{2,1} = \psi_1 = \xi_1$ dan $Q_2 = \{\xi_1\}$

  dan $J_{2,i} = \left(\overline{x_2} \setminus \{\psi_{j_i}\}\right)$ untuk i = 1,...,q_2 sehingga nilai i = 1 dan didapatkan:

  $J_{2,1} = J\left(\overline{x_2} \setminus \{\xi_1\}\right) = 9$ dan $J_2 = [9]^T$.

Dari perhitungan diatas didapat node-node keturunan dari *node* paling kanan sebanyak 1 *node* dan *tree*-nya dapat dilihat pada Gambar 3.3 berikut.



Gambar 3.3 Turunan node pada node *tree* level 2

Untuk menghindari duplikasi testing, maka fitur ψ_{j_i} dihilangkan dari himpunan fitur Ψ, yaitu sebagai berikut:

- $\Psi = \Psi \setminus Q_2$ dan $r = r - q_2$

  $\Psi = \phi$ $r = 1 - 1 = 0$

Kemudian dilanjutkan ke **Langkah 2**.

Pada langkah ini terjadi perhitungan nilai kriteria sebenarnya sebanyak 1 kali, sehingga jumlah evaluasi kriterianya menjadi 5.

Langkah 2

Telah diketahui :

$k = 2$, $q_2 = 1$, $r = 0$, $\Psi = \phi$, $\overline{x_2} = \{\xi_1, \xi_4, \xi_5\}$, $x^* = 3$

Selanjutnya diperiksa:

- IF $q_2 = 0$

  Telah diketahui bahwa $q_2 = 1$. Sehingga hal ini tidak memenuhi kondisi $q_2 = 0$.

- IF $J_{2,1} < x^*$

  Dari langkah sebelumnya telah diketahui bahwa $J_{2,1} = 9$ dan $x^* = 3$, sehingga tidak memenuhi kondisi yang diminta.

- Else

  $\overline{x_{2+1}} = \overline{x_2} \setminus \{Q_{2,1}\}$

  $\overline{x_3} = \{\xi_4, \xi_5\}$

  - If $k + 1 = D - d$

Karena $k+1=3$ dan $D-d=3$, maka memenuhi kondisi yang diminta.

Sehingga *leaf* ditemukan

Kemudian dilanjutkan ke **Langkah 5**.

Langkah 5

Nilai *bound* diupdate sebagai berikut:

$x^* = J_{2,1}$, sehingga $x^* = 9$

dan

$x = \overline{x_{2+1}}$, maka $x = \overline{x_3}$ dan sehingga $x = \{\xi_4, \xi_5\}$

Kemudian dilanjutkan ke **Langkah 2.**

Langkah 2

Telah diketahui :

$k = 2$, $q_2 = 1$, $r = 0$, $\Psi = \phi$, $\overline{x_2} = \{\xi_1, \xi_4, \xi_5\}$, $x^* = 9$

Selanjutnya diperiksa:

- IF $q_2 = 0$

  Telah diketahui bahwa $q_2 = 1$. Sehingga hal ini tidak memenuhi kondisi $q_2 = 0$.

- IF $J_{2,1} < x^*$

  Karena $J_{2,1} = 9$ dan $x^* = 9$, sehingga $J_{2,1} = x^*$. Kemudian dilanjutkan ke **Langkah 3**.

Langkah 3

Mengembalikan fitur $Q_{2,1}$ pada himpunan fitur Ψ, yaitu:

- $\Psi = \Psi \cup \{Q_{2,1}\}$ dan $r = r + 1$

  $\Psi = \{\xi_1\}$ $r = 0 + 1 = 1$

- $Q_2 = Q_2 \setminus \{Q_{2,1}\}$ dan $q_2 = q_2 - 1$

  $Q_2 = \phi$ $q_2 = 1 - 1 = 0$.

Kemudian dilanjutkan ke **Langkah 2.**

Langkah 2

Telah diketahui :

$k = 2$, $q_2 = 0$, $r = 1$, $\Psi = \{\xi_1\}$, $\overline{x_2} = \{\xi_1, \xi_4, \xi_5\}$, $x^* = 9$

Selanjutnya diperiksa:

- IF $q_2 = 0$

  Karena $q_2 = 0$. Sehingga memenuhi kondisi $q_2 = 0$, maka dilanjutkan ke **Langkah 4**.

Langkah 4

Backtracking:

- $k = k - 1$

  $k = 2 - 1$

  $k = 1$

  - If $k = -1$

    Karena $k = 1$, maka hal ini tidak memenuhi kondisi yang diminta.

  - Else

Mengembalikan fitur $Q_{1,1}$ pada himpunan kandidat.

Sehingga :

$\overline{X_1} = \overline{X_2} \cup \{Q_{1,1}\}$

$\overline{x_1} = \{\xi_1, \xi_2, \xi_4, \xi_5\}$

Kemudian dilanjutkan ke **Langkah 3.**

Langkah 3

Mengembalikan fitur $Q_{1,1}$ pada himpunan fitur Ψ, yaitu:

- $\Psi = \Psi \cup \{Q_{1,1}\}$ dan $r = r + 1$

  $\Psi = \{\xi_1, \xi_2\}$ $r = 1 + 1 = 2$

- $Q_1 = Q_1 \setminus \{Q_{1,1}\}$ dan $q_1 = q_1 - 1$

  $Q_1 = \phi$ $q_1 = 1 - 1 = 0$.

Kemudian dilanjutkan ke **Langkah 2.**

Langkah 2

Telah diketahui :

$k = 1$, $q_1 = 0$, $r = 2$, $\Psi = \{\xi_1, \xi_2\}$, $\overline{x_1} = \{\xi_1, \xi_2, \xi_4, \xi_5\}$, $x^* = 9$

Selanjutnya diperiksa:

- IF $q_1 = 0$

  Karena $q_1 = 0$. Sehingga memenuhi kondisi $q_1 = 0$, maka dilanjutkan ke **Langkah 4**.

Langkah 4

Backtracking:

- $k = k - 1$

$k = 1 - 1$

$k = 0$

- If $k = -1$

  Karena $k = 0$, maka hal ini tidak memenuhi kondisi yang diminta.

- Else

  Mengembalikan fitur $Q_{0,3}$ pada himpunan kandidat. Sehingga :

  $\overline{X_0} = \overline{X_1} \cup \{Q_{0,3}\}$

  $\overline{x_0} = \{\xi_1, \xi_2, \xi_3, \xi_4, \xi_5\}$

  Kemudian dilanjutkan ke **Langkah 3.**

Langkah 3

Mengembalikan fitur $Q_{0,3}$ pada himpunan fitur Ψ, yaitu:

- $\Psi = \Psi \cup \{Q_{0,3}\}$ dan $r = r + 1$

  $\Psi = \{\xi_1, \xi_2, \xi_3\}$ $r = 2 + 1 = 3$

- $Q_0 = Q_0 \setminus \{Q_{0,3}\}$ dan $q_0 = q_0 - 1$

  $Q_1 = \{\xi_5, \xi_4\}$ $q_0 = 3 - 1 = 2$.

Kemudian dilanjutkan ke **Langkah 2.**

Langkah 2

Telah diketahui :

$k = 0$, $q_0 = 2$, $r = 3$, $\Psi = \{\xi_1, \xi_2, \xi_3\}$, $\overline{x_0} = \{\xi_1, \xi_2, \xi_3, \xi_4, \xi_5\}$, $x^* = 9$

Selanjutnya diperiksa:

- IF $q_0 = 0$

  Telah diketahui bahwa $q_0 = 2$. Sehingga hal ini tidak memenuhi kondisi $q_0 = 0$.

- IF $J_{0,2} < x^*$

  Dari langkah sebelumnya telah diketahui bahwa $J_{0,2} = 11$ dan $x^* = 9$, sehingga tidak memenuhi kondisi yang diminta.

- Else

  $\overline{x_{0+1}} = \overline{x_0} \setminus \{Q_{0,2}\}$

  $\overline{x_1} = \{\xi_1, \xi_2, \xi_3, \xi_5\}$

  - If $k+1 = D-d$

    Karena $k+1 = 1$ dan $D-d = 3$, maka hal ini tidak memenuhi kondisi yang diminta.

  - Else

    $k+1 = 1$

    $k = 0+1$

    $k = 1$

Kemudian dilanjutkan ke **Langkah 1**.

Langkah 1

Telah diketahui

$k = 1$, $r = 3$, $\overline{x_1} = \{\xi_1, \xi_2, \xi_3, \xi_5\}$, $\Psi = \{\xi_1, \xi_2, \xi_3\}$, $x^* = 9$

maka:

- $q_1 = 3 - (5 - 2 - 1 - 1)$

$q_1 = 2$

- Untuk j = 1 sampai r, sehingga nilai j = 1,2,3

  $\psi_j \in \Psi$, dengan syarat : $A_{\psi_{j1}} \geq A_{\psi_{j2}} \geq A_{\psi_{j3}}$

  untuk j = 1, didapatkan:

  $\psi_1 = \xi_3$ sehingga

  $$A_3 = \frac{A_3 . S_3 + \left(J\left(\overline{x_1}\right) - J\left(\overline{x_1} \setminus \{\xi_3\}\right)\right)1}{S_3 + 1}$$

  $$A_3 = \frac{3.1 + (11-8).1}{1+1}$$

  $$A_3 = 3$$

  dan

  $$S_3 = S_3 + 1$$

  $$S_3 = 1 + 1$$

  $$S_3 = 2$$

  untuk j = 2, didapatkan:

  $\psi_2 = \xi_2$ sehingga

  $$A_2 = \frac{A_2 . S_2 + \left(J\left(\overline{x_1}\right) - J\left(\overline{x_1} \setminus \{\xi_2\}\right)\right)1}{S_2 + 1}$$

  $$A_2 = \frac{2.2 + (11-9).1}{2+1}$$

  $$A_2 = 2$$

  dan

  $$S_2 = S_2 + 1$$

  $$S_2 = 2 + 1$$

$S_2 = 3$

untuk j = 3, didapatkan:

$\psi_3 = \xi_1$ sehingga

$$A_1 = \frac{A_1 . S_1 + \left(J\left(\overline{x_1}\right) - J\left(\overline{x_1} \setminus \{\xi_1\}\right)\right)1}{S_1 + 1}$$

$$A_1 = \frac{1.3 + (11-10).1}{3+1}$$

$A_1 = 1$

dan

$S_1 = S_1 + 1$

$S_1 = 3 + 1$

$S_1 = 4$

- Kemudian dipilih 2 fitur dari 3 fitur yang ada pada Ψ, yaitu:

$Q_{1,i} = \psi_{J_i}$ untuk i = 1,...,q_1 sehingga nilai i = 1,2 dan didapatkan:

$Q_{1,1} = \psi_1 = \xi_3$

$Q_{1,2} = \psi_2 = \xi_2$

dan

$Q_1 = \{\xi_3, \xi_2\}$

dan $J_{1,i} = \left(\overline{x_1} \setminus \{\psi_{J_i}\}\right)$ untuk i = 1,...,q_1 sehingga nilai i = 1,2 dan didapatkan:

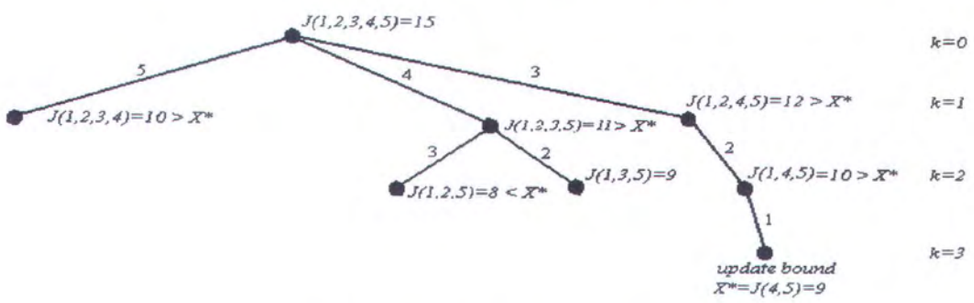
$J_{1,1} = J\left(\overline{x_1} \setminus \{\xi_3\}\right) = 8$

$J_{1,2} = J\left(\overline{x_1} \setminus \{\xi_2\}\right) = 9$

dan

$J_1 = [8,9]^T$.

Dari perhitungan diatas didapat node-node keturunan dari *node* pada level 1 sebanyak 2 *node* dan *tree*-nya dapat dilihat pada Gambar 3.4 berikut.



Gambar 3.4 Turunan node pada node *tree level* 1

Untuk menghindari duplikasi testing, maka fitur ψ_{j_i} dihilangkan dari himpunan fitur Ψ, yaitu sebagai berikut:

- $\Psi = \Psi \setminus Q_1$ dan $r = r - q_1$

  $\Psi = \{\xi_5\}$ $r = 3 - 2 = 1$

Kemudian dilanjutkan ke **Langkah 2**.

Pada langkah ini terjadi perhitungan nilai kriteria sebenarnya sebanyak 2 kali, sehingga jumlah evaluasi kriterianya menjadi 7.

Langkah 2

Telah diketahui :

$k = 1$, $q_1 = 2$, $r = 1$, $\Psi = \{\xi_5\}$, $\overline{x_1} = \{\xi_1, \xi_2, \xi_3, \xi_5\}$, $x^* = 9$

Selanjutnya diperiksa:

- IF $q_1 = 0$

  Telah diketahui bahwa $q_1 = 2$. Sehingga hal ini tidak memenuhi kondisi $q_1 = 0$.

- IF $J_{1,2} < x^*$

  Karena $J_{1,2} = 9$ dan $x^* = 9$, sehingga $J_{1,2} = x^*$. Kemudian dilanjutkan ke **Langkah 3**.

Langkah 3

Mengembalikan fitur $Q_{1,2}$ pada himpunan fitur Ψ, yaitu:

- $\Psi = \Psi \cup \{Q_{1,2}\}$ dan $r = r + 1$

  $\Psi = \{\xi_1, \xi_2\}$ $r = 1 + 1 = 2$

- $Q_1 = Q_1 \setminus \{Q_{1,2}\}$ dan $q_1 = q_1 - 1$

  $Q_1 = \{\xi_3\}$ $q_1 = 2 - 1 = 1$.

Kemudian dilanjutkan ke **Langkah 2.**

Langkah 2

Telah diketahui :

$k = 1$, $q_1 = 1$, $r = 2$, $\Psi = \{\xi_1, \xi_2\}$, $\overline{x_1} = \{\xi_1, \xi_2, \xi_3, \xi_5\}$, $x^* = 9$

Selanjutnya diperiksa:

- IF $q_1 = 0$

  Telah diketahui bahwa $q_1 = 1$. Sehingga hal ini tidak memenuhi kondisi $q_1 = 0$.

- IF $J_{1,1} < x^*$

  Karena $J_{1,1} = 8$ dan $x^* = 9$, sehingga memenuhi kondisi $J_{1,1} < x^*$.

  Kemudian dilanjutkan ke **Langkah 3**.

Langkah 3

Mengembalikan fitur $Q_{1,2}$ pada himpunan fitur Ψ, yaitu:

- $\Psi = \Psi \cup \{Q_{1,1}\}$ dan $r = r + 1$

  $\Psi = \{\xi_1, \xi_2, \xi_3\}$ $r = 2 + 1 = 3$

- $Q_1 = Q_1 \setminus \{Q_{1,1}\}$ dan $q_1 = q_1 - 1$

  $Q_1 = \phi$ $q_1 = 1 - 1 = 0$.

Kemudian dilanjutkan ke **Langkah 2.**

Langkah 2

Telah diketahui :

$k = 1$, $q_1 = 0$, $r = 3$, $\Psi = \{\xi_1, \xi_2, \xi_3\}$, $\overline{x_1} = \{\xi_1, \xi_2, \xi_3, \xi_5\}$, $x^* = 9$

Selanjutnya diperiksa:

- IF $q_1 = 0$

  Telah diketahui bahwa $q_1 = 0$. Sehingga memenuhi kondisi $q_1 = 0$.

  Kemudian dilanjutkan ke **Langkah 4**.

Langkah 4

Backtracking:

- $k = k - 1$

  $k = 1 - 1$

  $k = 0$

  - If $k = -1$

    Karena $k = 0$, maka hal ini tidak memenuhi kondisi yang diminta.

  - Else

Mengembalikan fitur $Q_{0,2}$ pada himpunan kandidat.

Sehingga :

$\overline{X_0} = \overline{X_1} \cup \{Q_{0,2}\}$

$\overline{x_0} = \{\xi_1, \xi_2, \xi_3, \xi_4, \xi_5\}$

Kemudian dilanjutkan ke **Langkah 3.**

Langkah 3

Mengembalikan fitur $Q_{0,2}$ pada himpunan fitur Ψ, yaitu:

- $\Psi = \Psi \cup \{Q_{0,2}\}$ dan $r = r + 1$

  $\Psi = \{\xi_1, \xi_2, \xi_3, \xi_4\}$ $r = 3 + 1 = 4$

- $Q_0 = Q_0 \setminus \{Q_{0,2}\}$ dan $q_0 = q_0 - 1$

  $Q_1 = \{\xi_5\}$ $q_0 = 2 - 1 = 1$.

Kemudian dilanjutkan ke **Langkah 2.**

Langkah 2

Telah diketahui :

$k = 0$, $q_0 = 1$, $r = 4$, $\Psi = \{\xi_1, \xi_2, \xi_3, \xi_4\}$, $\overline{x_0} = \{\xi_1, \xi_2, \xi_3, \xi_4, \xi_5\}$, $x^* = 9$

Selanjutnya diperiksa:

- IF $q_0 = 0$

  Telah diketahui bahwa $q_0 = 1$. Sehingga hal ini tidak memenuhi kondisi $q_0 = 0$.

- IF $J_{0,1} < x^*$

  Dari langkah sebelumnya telah diketahui bahwa $J_{0,2} = 10$ dan $x^* = 9$, sehingga tidak memenuhi kondisi yang diminta.

- Else

  $\overline{x_{0+1}} = \overline{x_0} \setminus \{Q_{0,1}\}$

  $\overline{x_1} = \{\xi_1, \xi_2, \xi_3, \xi_4\}$

  - If $k+1 = D-d$

    Karena $k+1=1$ dan $D-d=3$, maka hal ini tidak memenuhi kondisi yang diminta.

  - Else

    $k+1=1$

    $k=0+1$

    $k=1$

Kemudian dilanjutkan ke **Langkah 1**.

Langkah 1

Telah diketahui

$k=1$, $r=4$, $\overline{x_1} = \{\xi_1, \xi_2, \xi_3, \xi_4\}$, $\Psi = \{\xi_1, \xi_2, \xi_3, \xi_4\}$, $x^* = 9$

maka:

- $q_1 = 4-(5-2-1-1)$

  $q_1 = 3$

- Untuk j = 1 sampai r, sehingga nilai j = 1,2,3,4

  $\psi_j \in \Psi$, dengan syarat : $A_{\psi_{J1}} \geq A_{\psi_{J2}} \geq A_{\psi_{J3}} \geq A_{\psi_{J4}}$

  untuk j = 1, didapatkan:

  $\psi_1 = \xi_4$ sehingga

  $$A_4 = \frac{A_4 . S_4 + \left(J\left(\overline{x_1}\right) - J\left(\overline{x_1} \setminus \{\xi_4\}\right)\right)1}{S_4 + 1}$$

$$A_4 = \frac{4.1 + (10-6).1}{1+1}$$

$$A_4 = 4$$

dan

$$S_4 = S_4 + 1$$

$$S_4 = 1 + 1$$

$$S_4 = 2$$

untuk j = 2, didapatkan:

$\psi_2 = \xi_3$ sehingga

$$A_3 = \frac{A_3.S_3 + \left(J\left(\overline{x_1}\right) - J\left(\overline{x_1} \setminus \{\xi_3\}\right)\right).1}{S_3 + 1}$$

$$A_3 = \frac{3.2 + (10-7).1}{2+1}$$

$$A_3 = 3$$

dan

$$S_3 = S_3 + 1$$

$$S_3 = 2 + 1$$

$$S_3 = 3$$

untuk j = 3, didapatkan:

$\psi_3 = \xi_2$ sehingga

$$A_2 = \frac{A_2.S_2 + \left(J\left(\overline{x_1}\right) - J\left(\overline{x_1} \setminus \{\xi_2\}\right)\right).1}{S_2 + 1}$$

$$A_2 = \frac{2.3 + (10-8).1}{3+1}$$

$$A_2 = 2$$

dan

$$S_2 = S_2 + 1$$

$$S_2 = 3 + 1$$

$$S_2 = 4$$

untuk j = 4, didapatkan:

$\psi_4 = \xi_1$ sehingga

$$A_1 = \frac{A_1 . S_1 + \left(J\left(\overline{x_1}\right) - J\left(\overline{x_1} \setminus \{\xi_1\}\right)\right) 1}{S_1 + 1}$$

$$A_1 = \frac{1.4 + (10 - 9).1}{4 + 1}$$

$$A_1 = 1$$

dan

$$S_1 = S_1 + 1$$

$$S_1 = 4 + 1$$

$$S_1 = 5$$

- Kemudian dipilih 3 fitur dari 4 fitur yang ada pada Ψ, yaitu:

$Q_{1,i} = \psi_{j_i}$ untuk i = 1,...,q_1 sehingga nilai i = 1,2,3 dan didapatkan:

$$Q_{1,1} = \psi_1 = \xi_4$$

$$Q_{1,2} = \psi_2 = \xi_3$$

$$Q_{1,2} = \psi_2 = \xi_2$$

dan

$$Q_1 = \{\xi_4, \xi_3, \xi_2\}$$

dan $J_{1,i} = J(\overline{x_1} \setminus \{\psi_{j_i}\})$ untuk i = 1,...,q_1 sehingga nilai i = 1,2,3 dan didapatkan:

$$J_{1,1} = J(\overline{x_1} \setminus \{\xi_4\}) = 6$$

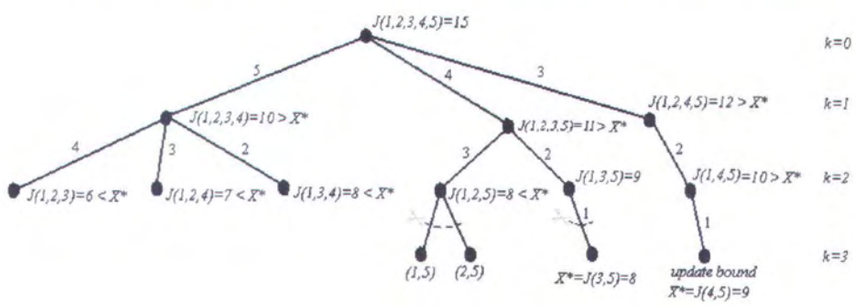
$$J_{1,2} = J(\overline{x_1} \setminus \{\xi_3\}) = 7$$

$$J_{1,3} = J(\overline{x_1} \setminus \{\xi_2\}) = 8$$

dan

$$J_1 = [6,7,8]^T .$$

Dari perhitungan diatas didapat node-node keturunan dari *node* pada level 1 sebanyak 3 *node* dan *tree*-nya dapat dilihat pada Gambar 3.5 berikut.



Gambar 3.5 Turunan node pada node *tree level* 1

Untuk menghindari duplikasi testing, maka fitur ψ_{j_i} dihilangkan dari himpunan fitur Ψ, yaitu sebagai berikut:

- $\Psi = \Psi \setminus Q_1$ dan $r = r - q_1$

  $\Psi = \{\xi_1\}$ $r = 4 - 3 = 1$

Kemudian dilanjutkan ke **Langkah 2**.

Langkah 2

Telah diketahui :

$k = 1$, $q_1 = 3$, $r = 1$, $\Psi = \{\xi_1\}$, $\overline{x_1} = \{\xi_1, \xi_2, \xi_3, \xi_4\}$, $x^* = 9$

Selanjutnya diperiksa:

- IF $q_1 = 0$

  Telah diketahui bahwa $q_1 = 3$. Sehingga hal ini tidak memenuhi kondisi $q_1 = 0$.

- IF $J_{1,3} < x^*$

  Karena $J_{1,3} = 8$ dan $x^* = 9$, sehingga memenuhi kondisi $J_{1,3} < x^*$.

  Kemudian dilanjutkan ke **Langkah 3**.

Langkah 3

Mengembalikan fitur $Q_{1,3}$ pada himpunan fitur Ψ, yaitu:

- $\Psi = \Psi \cup \{Q_{1,3}\}$ dan $r = r + 1$

  $\Psi = \{\xi_1, \xi_2\}$ $r = 1 + 1 = 2$

- $Q_1 = Q_1 \setminus \{Q_{1,3}\}$ dan $q_1 = q_1 - 1$

  $Q_1 = \{\xi_4, \xi_3\}$ $q_1 = 3 - 1 = 2$.

Kemudian dilanjutkan ke **Langkah 2.**

Langkah 2

Telah diketahui :

$k = 1$, $q_1 = 2$, $r = 2$, $\Psi = \{\xi_1, \xi_2\}$, $\overline{x_1} = \{\xi_1, \xi_2, \xi_3, \xi_4\}$, $x^* = 9$

Selanjutnya diperiksa:

- IF $q_1 = 0$

  Telah diketahui bahwa $q_1 = 2$. Sehingga hal ini tidak memenuhi kondisi $q_1 = 0$.

- IF $J_{1,2} < x^*$

Karena $J_{1,2}=7$ dan $x^*=9$, sehingga memenuhi kondisi $J_{1,2}<x^*$.

Kemudian dilanjutkan ke **Langkah 3**.

Langkah 3

Mengembalikan fitur $Q_{1,2}$ pada himpunan fitur Ψ, yaitu:

- $\Psi=\Psi\cup\{Q_{1,2}\}$ dan $r=r+1$

  $\Psi=\{\xi_1,\xi_2,\xi_3\}$ $r=2+1=3$

- $Q_1=Q_1\setminus\{Q_{1,2}\}$ dan $q_1=q_1-1$

  $Q_1=\{\xi_4\}$ $q_1=2-1=1$.

Kemudian dilanjutkan ke **Langkah 2.**

Langkah 2

Telah diketahui :

$k=1$, $q_1=1$, $r=3$, $\Psi=\{\xi_1,\xi_2,\xi_3\}$, $\overline{x_1}=\{\xi_1,\xi_2,\xi_3,\xi_4\}$, $x^*=9$

Selanjutnya diperiksa:

- IF $q_1=0$

  Telah diketahui bahwa $q_1=1$. Sehingga hal ini tidak memenuhi kondisi $q_1=0$.

- IF $J_{1,1}<x^*$

  Karena $J_{1,1}=6$ dan $x^*=9$, sehingga memenuhi kondisi $J_{1,1}<x^*$.

  Kemudian dilanjutkan ke **Langkah 3**.

Langkah 3

Mengembalikan fitur $Q_{1,1}$ pada himpunan fitur Ψ, yaitu:

- $\Psi=\Psi\cup\{Q_{1,1}\}$ dan $r=r+1$

$\Psi = \{\xi_1, \xi_2, \xi_3, \xi_4\}$ $r = 3+1 = 4$

- $Q_1 = Q_1 \setminus \{Q_{1,1}\}$ dan $q_1 = q_1 - 1$

  $Q_1 = \phi$ $q_1 = 1-1 = 0$.

Kemudian dilanjutkan ke **Langkah 2.**

Langkah 2

Telah diketahui :

$k = 1$, $q_1 = 0$, $r = 4$, $\Psi = \{\xi_1, \xi_2, \xi_3, \xi_4\}$, $\overline{x_1} = \{\xi_1, \xi_2, \xi_3, \xi_4\}$, $x^* = 9$

Selanjutnya diperiksa:

- IF $q_1 = 0$

  Telah diketahui bahwa $q_1 = 0$. Sehingga memenuhi kondisi $q_1 = 0$.

  Kemudian dilanjutkan ke **Langkah 4**.

Langkah 4

Backtracking:

- $k = k - 1$

  $k = 1 - 1$

  $k = 0$

  - If $k = -1$

    Karena $k = 0$, maka hal ini tidak memenuhi kondisi yang diminta.

  - Else

    Mengembalikan fitur $Q_{0,1}$ pada himpunan kandidat. Sehingga :

$$\overline{X_0} = \overline{X_1} \cup \{Q_{0,1}\}$$

$$\overline{x_0} = \{\xi_1, \xi_2, \xi_3, \xi_4, \xi_5\}$$

Kemudian dilanjutkan ke **Langkah 3.**

Langkah 3

Mengembalikan fitur $Q_{0,1}$ pada himpunan fitur Ψ, yaitu:

- $\Psi = \Psi \cup \{Q_{0,1}\}$ dan $r = r + 1$

  $\Psi = \{\xi_1, \xi_2, \xi_3, \xi_4, \xi_5\}$ $r = 4 + 1 = 5$

- $Q_0 = Q_0 \setminus \{Q_{0,1}\}$ dan $q_0 = q_0 - 1$

  $Q_0 = \phi$ $q_0 = 1 - 1 = 0$.

Kemudian dilanjutkan ke **Langkah 2.**

Langkah 2

Telah diketahui :

$k = 0$, $q_0 = 0$, $r = 5$, $\Psi = \{\xi_1, \xi_2, \xi_3, \xi_4, \xi_5\}$, $\overline{x_0} = \{\xi_1, \xi_2, \xi_3, \xi_4, \xi_5\}$, $x^* = 9$

Selanjutnya diperiksa:

- IF $q_0 = 0$

  Telah diketahui bahwa $q_0 = 0$. Sehingga memenuhi kondisi $q_0 = 0$.

  Kemudian dilanjutkan ke **Langkah 4**.

Langkah 4

Backtracking:

- $k = k - 1$

  $k = 0 - 1$

  $k = -1$

- If $k = -1$

  Karena $k = -1$, maka memenuhi kondisi yang diminta. Sehingga algoritma telah selesai.

Hasil penelusuran *tree* secara keseluruhan dapat dilihat pada Gambar 3.5 dan menghasilkan jumlah evaluasi kriteria sebanyak 13 dengan nilai *bound* adalah 9 dan fitur yang dipilih adalah fitur 4 dan fitur 5.

3.2 *FAST BRANCH & BOUND*

Algoritma *Fast Branch & Bound* (FBB) merupakan algoritma *Branch & Bound* dengan menggunakan suatu mekanisme prediksi yang lebih baik daripada algoritma *Branch & Bound Partial Prediction*. Algoritma *Fast Branch & Bound* (FBB) ini lebih mengurangi jumlah perhitungan fungsi kriteria dalam *internal node* pohon pencarian. Algoritma menggunakan informasi dari nilai kriteria turunan untuk prediksi nilai kriteria selanjutnya. Perhitungan nilai kriteria dan prediksi nilai kriteria sama-sama diperlukan dalam algoritma ini. Keduanya digunakan untuk mengurutkan *node-node* pohon, memeriksa *subtree* apakah memungkinkan untuk di-*cut off*, dan lain sebagainya. Algoritma ini hanya mengijinkan prediksi jika keadaannya memungkinkan, yaitu jika *node* yang diprediksi bukan *leaves* dan bukan merupakan *node* yang men-*trigger* sebuah *node* untuk di-*cut-off*.

Setiap prediksi harus diperiksa dan dibandingkan dengan nilai *bound*. Jika prediksi nilai kriteria lebih besar dari nilai *bound*, maka nilai sebenarnya tidak akan lebih rendah dari nilai *bound*. Algoritma tidak boleh meng-*cut off subtree*

yang berhubungan dan algoritma melanjutkan membentuk level pohon selanjutnya. Namun, jika prediksi nilai kriteria sama dengan nilai *bound*, dalam kondisi ini ada kemungkinan nilai kriteria sebenarnya lebih kecil dari nilai *bound*, maka nilai kriteria sebenarnya harus dihitung. Hanya jika nilai kriteria sebenarnya membuktikan bahwa lebih rendah daripada nilai *bound* saat itu, maka *subtree* boleh di-*cut off*.

Prediksi yang digunakan dalam algoritma ini tidak mempengaruhi keoptimalan hasil yang didapat. Prediksi yang tidak akurat tidak akan memperburuk pada pembentukan *subtree* yang akan di-*cut off*. Algoritma menggunakan prediksi ini sesering mungkin, karena hal ini bisa menghemat waktu.

Algoritma FBB ini juga menggunakan vektor kontribusi dan vektor kounter seperti yang digunakan dalam algoritma BBPP. Sebelum setiap kriteria dievaluasi, rekord pada vektor kounter diperiksa lebih dulu apakah cocok untuk masukan selanjutnya. Jika mekanisme prediksi ini telah mendapatkan informasi yang cukup (nilai kounter > spesifikasi minimum), maka nilai kriteria akan diprediksi, dan jika sebaliknya, maka nilai kriteria sebenarnya yang akan dihitung.

Sangat penting membedakan antara nilai prediksi dan nilai kriteria sebenarnya. Oleh karena itu, dalam algoritma ini digunakan sebuah vektor dari nilai tipe (didefinisikan dengan vektor tipe). Vektor tipe ini untuk menyimpan tipe *node-node* pada *level* pohon saat itu. Nilai tipe yang digunakan dalam vektor tipe ini adalah "P", jika nilai kriterianya diprediksi, dan "C", jika nilai kriterianya dihitung.

Notasi-notasi yang digunakan dalam algoritma *Fast Branch & Bound* sama dengan notasi yang digunakan dalam algoritma BBPP, namun ada sedikit tambahan notasi yang digunakan, yaitu:

a) $\delta \geq 1$ yang merupakan jumlah evaluasi minimum

b) $\gamma \geq 0$ yang merupakan optimism

c) $T_k = [T_{k,1}, T_{k,2}, ...T_{k,q_k}]^T$, $T_{k,i} \in \{"C", "P"\}$ untuk $i = 1,...,q_k$ yang merupakan vektor tipe dari nilai kriteria

d) $V = [v_1, v_2, ..., v_{q_k}]^T$ yang merupakan vektor terurut untuk sementara.

Ketika algoritma menghilangkan beberapa fitur y_i dari himpunan kandidat fitur yang ada dan menghitung nilai kriteria sebenarnya yang berhubungan, yaitu $J(\overline{x_k} \setminus \{y_i\})$, pada level pohon ke-k dan jika nilai sebelumnya $J(\overline{x_k}) \equiv J(\overline{x_{k-1}} \setminus \{y_j\})$ telah dihitung (ditunjukkan oleh $T_{k-1,y_j} = "C"$), maka A_{y_i} dan S_{y_i} diupdate dengan rumus dibawah ini.

$$A_{y_i} = \frac{A_{y_i}.S_{y_i} + J_{k-1,y_j} - J(\overline{x_k} \setminus \{y_i\})}{S_{y_i} + 1} \quad \text{.....(3.21)}$$

dan

$$S_{y_i} = S_{y_i} + 1 \quad \text{.....(3.22)}$$

3.2.1 Algoritma *Fast Branch & Bound*

Algoritma ini mempunyai tahap-tahap yang sama dengan algoritma BBPP. Algoritma ini tidak bisa digunakan dengan bentuk fungsi kriteria yang rekursi, dimana perhitungan nilai $J(\overline{x_k})$ memerlukan informasi dari nilai sebelumnya

$J(\overline{x_{k-1}})$. Algoritma FBB mengurangi evaluasi fungsi kriteria pada banyak *internal node*, nilai kriteria yang berhubungan tidak dapat dihitung secara rekursi pada node turunan selanjutnya. Oleh karena itu, algoritma FBB ini akan lebih efektif bila digunakan dengan fungsi kriteria yang tidak rekursi pada kompleksitas perhitungan yang lebih tinggi.

Langkah-langkah dari algoritma FBB adalah sebagai berikut [SPK04]:

Inisialisasi:

$k=0$, $\overline{\chi}_0 = Y$, $\Psi = Y$, $r = D$, $x^* = \sim$

Langkah 1 Memilih keturunan dari *current node* untuk membentuk level pohon yang berhubungan

- $q_k = r-(D-d-k-1)$ .....(3.23)
- dibentuk Q_k , J_k , T_k sebagai berikut:

  Untuk tiap fitur $\psi_j \in \Psi$ dan $j = 1,...,r$

  - *If* $(k+1 < D-d)$ *and* $S_{\psi_j} > \delta$

    Hal ini berarti prediksi diijinkan dan kemudian

    $v_j = J_{k-1,q_{k-1}} - A_{\psi_j}$ .....(3.24).

  - *Else*

    Maka nilai kriteria sebenarnya yang harus dihitung dan

    $v_j = J(\overline{x_k} \setminus \{\psi_j\})$ .....(3.25)

  Setelah semua nilai v_j diperoleh, maka selanjutnya diurutkan secara ascending, yaitu :

  $v_{j_1} \le v_{j_2} \le ... \le v_{j_r}$ .....(3.26)

- Untuk nilai $i = 1,...,q_k$, maka:

  $Q_{k,i} = \psi_{j_i}$ .....(3.27)

  dan

  $J_{k,i} = v_{j_i}$ .....(3.28),

  jika v_{j_i} merupakan sebuah rekord yang nilainya dihitung dan

  $J_{k,i} = J_{k-1,q_{k-1}} - \gamma.A_{\psi_{j_i}}$ .....(3.29),

  jika v_{j_i} merupakan sebuah rekord yang nilainya diprediksi.

  $T_{k,i} = "C"$ .....(3.30),

  jika v_{j_i} merupakan sebuah rekord yang nilainya dihitung dan

  $T_{k,i} = "P"$ .....(3.31)

  jika v_{j_i} merupakan sebuah rekord yang nilainya diprediksi

- Untuk menghindari evaluasi ganda, maka :

  $\Psi = \Psi \setminus Q_k$ .....(3.32)

  dan

  $r = r - q_k$ .....(3.33)

Langkah 2 Memilih turunan node paling kanan

- If $q_k = 0$, maka semua turunan telah diperiksa dan dilanjutkan ke **Langkah 4** (*backtracking*)
- If $(T_{k,q_k} = "P")\ AND\ J_{k,q_k} < x^*$, maka nilai sebenarnya yang dihitung, yaitu:

  $J_{k,q_k} = J(\overline{x_k} \setminus \{Q_{k,q_k}\})$ .....(3.34)

dan

$T_{k,q_k} = "C"$ .....(3.35)

- If $(T_{k,q_k} = "C")\ AND\ J_{k,q_k} < x^*$, maka dilanjutkan ke **Langkah 3**.
- Else

$\overline{X_{k+1}} = \overline{X_k} \setminus \{Q_{k,q_k}\}$ .....(3.36)

- *If* $k+1 = D-d$, maka telah mencapai satu leaf dan dilanjutkan ke **Langkah 5**
- *Else*

Menuju ke level selanjutnya dengan

$k = k+1$ .....(3.37)

dan dilanjutkan ke **Langkah 1**

Untuk langkah 3, 4 dan 5 sama dengan langkah 3, 4 dan 5 pada algoritma IBB, yaitu sebagai berikut:

Langkah 3 Turunan node kemungkinan di-*cut-off*

Mengembalikan fitur $Q_{k,qk}$ ke himpunan fitur yang ada pada pembuatan tree, yaitu

- $\Psi = \Psi \cup \{Q_{k,q_k}\}$ .....(3.38)

dan

$r = r+1$ .....(3.39),

- $Q_k = Q_k \setminus \{Q_{k,q_k}\}$ .....(3.40)

dan

$q_k = q_k - 1$ .....(3.41)

kemudian dilanjutkan dengan node sebelah kirinya, dan dilanjutkan ke **Langkah 2**.

Langkah 4 *Backtracking*:

- $k = k - 1$ .....(3.42)
  - *If* $k = -1$, maka semua *tree* telah diperiksa sampai selesai, dan algoritma selesai.
  - *Else*

    Mengembalikan fitur Q_{k,q_k} pada himpunan kandidat, sehingga :

    $\overline{X_k} = \overline{X_{k+1}} \cup \{Q_{k,q_k}\}$ .....(3.43)

    dan dilanjutkan ke **Langkah 3.**

Langkah 5 *Update* nilai *bound*:

Nilai *bound* di-*update*:

- $x^* = J_{k,q_k}$ .....(3.44)
- fitur terbaik disimpan dalam

  $X = \overline{X_{k+1}}$ .....(3.45)

  dan dilanjutkan ke **Langkah 2**.

3.2.2 Contoh Algoritma *Fast Branch & Bound*

Permasalahan yang diambil yaitu misalkan ada himpunan fitur awal dengan jumlah fiturnya sebanyak 5 fitur. Dari himpunan fitur awal ini akan dipilih 2 fitur saja dengan fungsi kriterianya adalah $J(\bar{x}) = \sum_{\xi_i \in x} i$.

Berdasarkan pada permasalah diatas maka penyelesaiannya menggunakan algoritma FBB adalah sebagai berikut:

Inisialisasi awal:

$D=5\rightarrow\{f_1,f_2,f_3,f_4,f_5\}$, $d=2$, $J(\overline{x})=\sum_{\xi_i\in x} i$, $k=0$, $\Psi=Y$, $r=D$, $x^*=\sim$, $\delta=0$,

$\gamma=0$, $\overline{x_0}=\{\xi_j|j=1,2,3,4,5\}\rightarrow\overline{x_0}=\{\xi_1,\xi_2,\xi_3,\xi_4,\xi_5\}$, $x=\{x_j|j=1,2\}\rightarrow x=\{x_1,x_2\}$,

$Y=\overline{x_0}$

Langkah 1 Memilih keturunan dari *current* node untuk membentuk level pohon yang berhubungan

Telah diketahui :

$k=0$, $r=5$, dan *current* node adalah *root*

maka:

- $q_k=r-(D-d-k-1)$

  $q_0=5-(5-2-0-1)$

  $q_0=3$

- Untuk j = 1 sampai r sehingga nilai j = 1, 2, 3, 4, 5

  $\psi_j\in\Psi$

  untuk j = 1, didapatkan:

  $\psi_1=\xi_1$

  - *If* $(k+1<D-d)$ *and* $S_1>\delta$

    Telah diketahui sebelumnya bahwa $k+1=1$ dan $D-d=3$ maka hal ini memenuhi kondisi $(k+1<D-d)$. Namun,

karena $S_1 = 0$ dan $\delta = 0$ maka tidak memenuhi kondisi $S_1 > \delta$.

- *Else*

  Maka nilai kriterianya harus dihitung ("C"), yaitu:

  $v_1 = J(\overline{x_0} \setminus \{\xi_1\})$

  $v_1 = J(2,3,4,5)$

  $v_1 = 14$

untuk j = 2, didapatkan:

$\psi_2 = \xi_2$

- *If* $(k+1 < D-d)$ *and* $S_2 > \delta$

  Telah diketahui sebelumnya bahwa $k+1 = 1$ dan $D-d = 3$ maka hal ini memenuhi kondisi $(k+1 < D-d)$. Namun, karena $S_2 = 0$ dan $\delta = 0$ maka tidak memenuhi kondisi $S_2 > \delta$.

- *Else*

  Maka nilai kriterianya harus dihitung ("C"), yaitu:

  $v_2 = J(\overline{x_0} \setminus \{\xi_2\})$

  $v_2 = J(1,3,4,5)$

  $v_2 = 13$

untuk j = 3, didapatkan:

$\psi_3 = \xi_3$

- *If* $(k+1 < D-d)$ *and* $S_3 > \delta$

Telah diketahui sebelumnya bahwa $k+1=1$ dan $D-d=3$ maka hal ini memenuhi kondisi $(k+1<D-d)$. Namun, karena $S_3=0$ dan $\delta=0$ maka tidak memenuhi kondisi $S_3>\delta$.

- *Else*

  Maka nilai kriterianya harus dihitung ("C"), yaitu:

  $v_3=J(\overline{x_0}\setminus\{\xi_3\})$

  $v_3=J(1,2,4,5)$

  $v_3=12$

untuk j = 4, didapatkan:

$\psi_4=\xi_4$

- *If* $(k+1<D-d)$ *and* $S_4>\delta$

  Telah diketahui sebelumnya bahwa $k+1=1$ dan $D-d=3$ maka hal ini memenuhi kondisi $(k+1<D-d)$. Namun, karena $S_4=0$ dan $\delta=0$ maka tidak memenuhi kondisi $S_4>\delta$.

- *Else*

  Maka nilai kriterianya harus dihitung ("C"), yaitu:

  $v_4=J(\overline{x_0}\setminus\{\xi_4\})$

  $v_4=J(1,2,3,5)$

  $v_4=11$

untuk j = 5, didapatkan:

$\psi_5 = \xi_5$

- *If* $(k+1 < D-d)$ *and* $S_5 > \delta$

  Telah diketahui sebelumnya bahwa $k+1=1$ dan $D-d=3$ maka hal ini memenuhi kondisi $(k+1 < D-d)$. Namun, karena $S_5 = 0$ dan $\delta = 0$ maka tidak memenuhi kondisi $S_5 > \delta$.

- *Else*

  Maka nilai kriterianya harus dihitung ("C"), yaitu:

  $v_5 = J(\overline{x_0} \setminus \{\xi_5\})$

  $v_5 = J(1,2,3,4)$

  $v_5 = 10$

kemudian v_j diurutkan secara ascending, maka didapatkan:

$\psi_1 = \xi_5$ dan $v_1 = 10$

$\psi_2 = \xi_4$ dan $v_2 = 11$

$\psi_3 = \xi_3$ dan $v_3 = 12$

$\psi_4 = \xi_2$ dan $v_4 = 13$

$\psi_5 = \xi_1$ dan $v_5 = 14$

sehingga $v_1 \le v_2 \le v_3 \le v_4 \le v_5$

- untuk i = 1,...,q_0 sehingga nilai i = 1,2,3, dan didapatkan:

untuk i = 1

$Q_{0,1} = \psi_1 = \xi_5$

$J_{0,1} = v_1 = 10$

$T_{0,1} = "C"$

untuk i = 2

$Q_{0,2} = \psi_2 = \xi_4$

$J_{0,2} = v_2 = 11$

$T_{0,2} = "C"$

untuk i = 3

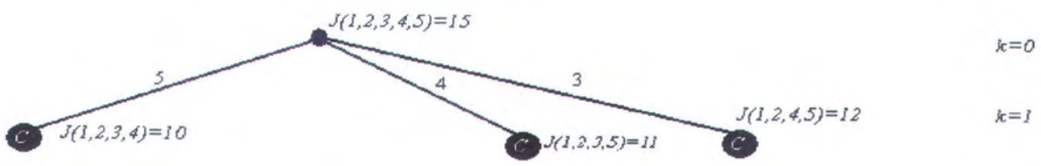
$Q_{0,3} = \psi_3 = \xi_3$

$J_{0,3} = v_3 = 12$

$T_{0,3} = "C"$

dan didapatkan $Q_0 = \{\xi_5, \xi_4, \xi_3\}$, $J_0 = [10 \quad 11 \quad 12]^T$, dan $T_0 = ["C" \quad "C" \quad "C"]^T$

Dari perhitungan diatas didapat *node-node* keturunan dari *root* sebanyak 3 dan *tree*-nya dapat dilihat pada Gambar 3.6 berikut.



Gambar 3.6 Turunan *node* pada *node tree level* 0

Untuk menghindari duplikasi testing, maka fitur ψ_{j_i} dihilangkan dari himpunan fitur Ψ, yaitu sebagai berikut:

- $\Psi = \Psi \setminus Q_0$ dan $r = r - q_0$

  $\Psi = \{\xi_1, \xi_2\}$ $r = 5 - 3 = 2$

Kemudian dilanjutkan ke **Langkah 2**.

Langkah 2 Memeriksa turunan node paling kanan

Telah diketahui :

$k=0,\quad q_0=3,\quad r=2,\quad \Psi=\{\xi_1,\xi_2\},\quad \overline{X_0}=\{\xi_1,\xi_2,\xi_3,\xi_4,\xi_5\},\quad x^*=\sim,$

$Q_0=\{\xi_5,\xi_4,\xi_3\},\ J_0=[10\quad 11\quad 12]^T,\ T_0=["C"\quad "C"\quad "C"]^T$

Selanjutnya diperiksa:

- *If* $q_0=0$

  Telah diketahui bahwa $q_0=3$. Sehingga hal ini tidak memenuhi kondisi $q_0=0$.

- If $(T_{0,3}="P")\ AND\ J_{0,3}<x^*$

  Telah diketahui bahwa $T_{0,3}="C"$ dan $J_{0,3}>x^*$, maka hal ini tidak memenuhi kondisi yang diminta.

- If $(T_{0,3}="C")\ AND\ J_{0,3}<x^*$

  Telah diketahui bahwa $T_{0,3}="C"$ dan $J_{0,3}>x^*$, maka hal ini tidak memenuhi kondisi yang diminta.

- Else

  $\overline{X_{0+1}}=\overline{X_0}\setminus\{Q_{0,3}\}$

  $\overline{x_1}=\{\xi_1,\xi_2,\xi_4,\xi_5\}$

  - *If* $k+1=D-d$

    Telah diketahui bahwa $k+1=1$ dan $D-d=3$, sehingga hal ini tidak memenuhi kondisi yang diminta.

  - *Else*

    $k=k+1$

$$k = 0 + 1$$

$$k = 1$$

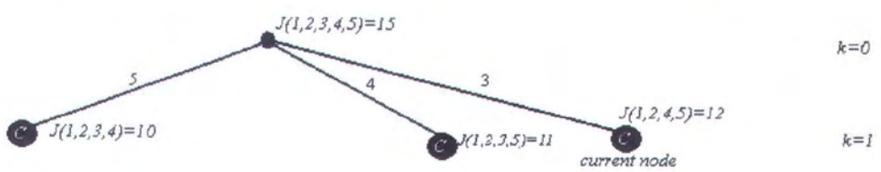
dan dilanjutkan ke **Langkah 1**

Langkah 1 Memilih keturunan dari *current* node untuk membentuk level pohon yang berhubungan

Telah diketahui

$k = 1$, $r = 2$, $\overline{x_1} = \{\xi_1, \xi_2, \xi_4, \xi_5\}$, $\Psi = \{\xi_1, \xi_2\}$, $x^* = \sim$

pada gambar 3.7 dibawah ini menunjukkan posisi *current* node.



Gambar 3.7 ***Node*** **dengan** $J(1,2,4,5)$ **sebagai** ***current node***

maka:

- $q_1 = 2 - (5 - 2 - 1 - 1)$

  $q_1 = 1$

- Untuk j = 1 sampai r sehingga nilai j = 1, 2

  $\psi_j \in \Psi$

  untuk j = 1, didapatkan:

  $\psi_1 = \xi_2$

  - *If* $(k+1 < D-d)$ *and* $S_2 > \delta$

    Telah diketahui sebelumnya bahwa $k+1 = 2$ dan $D-d = 3$ maka hal ini memenuhi kondisi $(k+1 < D-d)$. Namun,

karena $S_2 = 0$ dan $\delta = 0$ maka tidak memenuhi kondisi $S_2 > \delta$.

- *Else*

  Maka nilai kriterianya harus dihitung ("C"), yaitu:

  $v_1 = J(\overline{x_1} \setminus \{\xi_2\})$

  $v_1 = J(1,4,5)$

  $v_1 = 10$

untuk j = 2, didapatkan:

$\psi_2 = \xi_1$

- *If* $(k+1 < D-d)$ *and* $S_1 > \delta$

  Telah diketahui sebelumnya bahwa $k+1 = 2$ dan $D-d = 3$ maka hal ini memenuhi kondisi $(k+1 < D-d)$. Namun, karena $S_1 = 0$ dan $\delta = 0$ maka tidak memenuhi kondisi $S_1 > \delta$.

- *Else*

  Maka nilai kriterianya harus dihitung ("C"), yaitu:

  $v_2 = J(\overline{x_1} \setminus \{\xi_1\})$

  $v_2 = J(2,4,5)$

  $v_3 = 11$

sehingga $v_1 \leq v_2$

- untuk i = 1,...,q_1 sehingga nilai i = 1 dan didapatkan:

  untuk i = 1

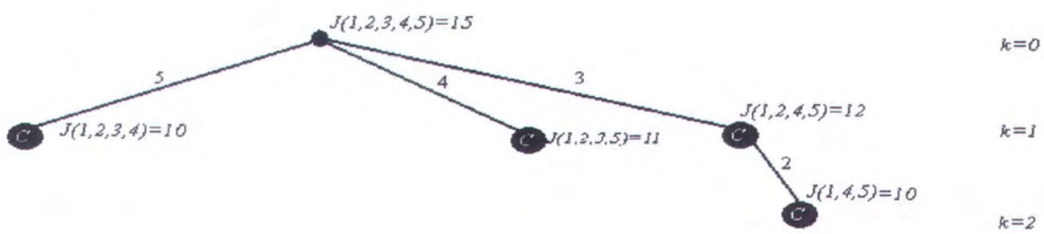
$$Q_{1,1} = \psi_1 = \xi_2$$

$$J_{1,1} = v_1 = 10$$

$$T_{1,1} = "C"$$

dan didapatkan $Q_1 = \{\xi_2\}$, $J_1 = [10]^T$, dan $T_1 = ["C"]^T$

Dari perhitungan diatas didapat node-node keturunan dari *current* node sebanyak 1 dan *tree*-nya dapat dilihat pada Gambar 3.8 berikut.



Gambar 3.8 Turunan *node* pada *node tree level* 1

Untuk menghindari duplikasi testing, maka fitur ψ_{j_i} dihilangkan dari himpunan fitur Ψ, yaitu sebagai berikut:

- $\Psi = \Psi \setminus Q_1$ dan $r = r - q_1$

  $\Psi = \{\xi_1\}$ $r = 2 - 1 = 1$

Kemudian dilanjutkan ke **Langkah 2**.

Update A_3 dan S_3

A_3 dan S_3 diupdate sebagai berikut:

$$A_3 = \frac{A_3.S_3 + J_{-1,y_j} - J(\overline{x_1} \setminus \{\xi_3\})}{S_3 + 1}$$

$$A_3 = \frac{0.0 + 15 - 12}{1}$$

$$A_3 = 3$$

dan

$S_3 = S_3 + 1$

$S_3 = 0 + 1 = 1$

Langkah 2 Memeriksa turunan node paling kanan

Telah diketahui

$k = 1,\ r = 1,\ \overline{x_1} = \{\xi_1, \xi_2, \xi_4, \xi_5\},\ \Psi = \{\xi_1\},\ x^* = \sim$

Selanjutnya diperiksa:

- *If* $q_1 = 0$

  Telah diketahui bahwa $q_1 = 1$. Sehingga hal ini tidak memenuhi kondisi $q_0 = 0$.

- If $(T_{1,1} = "P")\ AND\ J_{1,1} < x^*$

  Telah diketahui bahwa $T_{1,1} = "C"$ dan $J_{1,1} > x^*$, maka hal ini tidak memenuhi kondisi yang diminta.

- If $(T_{1,1} = "C")\ AND\ J_{1,1} < x^*$

  Telah diketahui bahwa $T_{1,1} = "C"$ dan $J_{1,1} > x^*$, maka hal ini tidak memenuhi kondisi yang diminta.

- Else

  $\overline{X_{1+1}} = \overline{X_1} \setminus \{Q_{1,1}\}$

  $\overline{x_2} = \{\xi_1, \xi_4, \xi_5\}$

  - *If* $k + 1 = D - d$

Telah diketahui bahwa $k+1=2$ dan $D-d=3$, sehingga hal ini tidak memenuhi kondisi yang diminta.

- *Else*

$$k = k+1$$

$$k = 1+1$$

$$k = 2$$

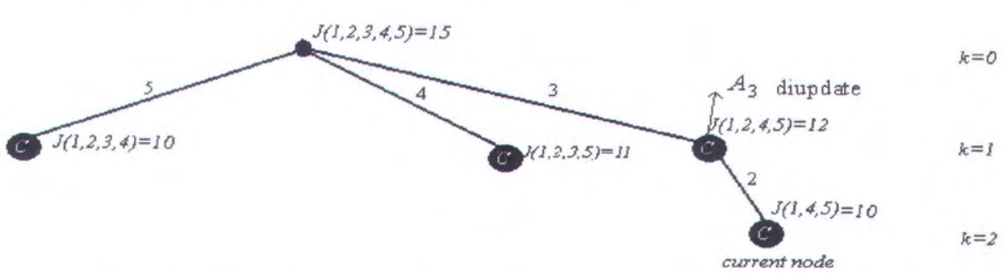
dan dilanjutkan ke **Langkah 1**

Langkah 1 Memilih keturunan dari *current* node untuk membentuk *level* pohon yang berhubungan

Telah diketahui

$k=2,\ r=1,\ \overline{x_2}=\{\xi_1,\xi_4,\xi_5\},\ \Psi=\{\xi_1\},\ x^*=\sim$

pada Gambar 3.9 dibawah ini menunjukkan posisi *current* node.



Gambar 3.9 ***Node*** **dengan** $J(1,4,5)$ **sebagai** ***current node***

maka:

- $q_2 = 1-(5-2-2-1)$

  $q_2 = 1$

- Untuk j = 1 sampai r sehingga nilai j = 1

  $\psi_j \in \Psi$

untuk j = 1, didapatkan:

$\psi_1 = \xi_1$

- *If* $(k+1 < D-d)$ *and* $S_1 > \delta$

  Telah diketahui sebelumnya bahwa $k+1=3$ dan $D-d=3$ maka hal ini tidak memenuhi kondisi $(k+1 < D-d)$. Begitu juga untuk nilai $S_1 = 0$ dan $\delta = 0$ maka tidak memenuhi kondisi $S_1 > \delta$.

- *Else*

  Maka nilai kriterianya harus dihitung ("C"), yaitu:

  $v_1 = J\left(\overline{x_2} \setminus \{\xi_1\}\right)$

  $v_1 = J(4,5)$

  $v_1 = 9$

- untuk i = 1,...,q_2 sehingga nilai i = 1 dan didapatkan:

  untuk i = 1

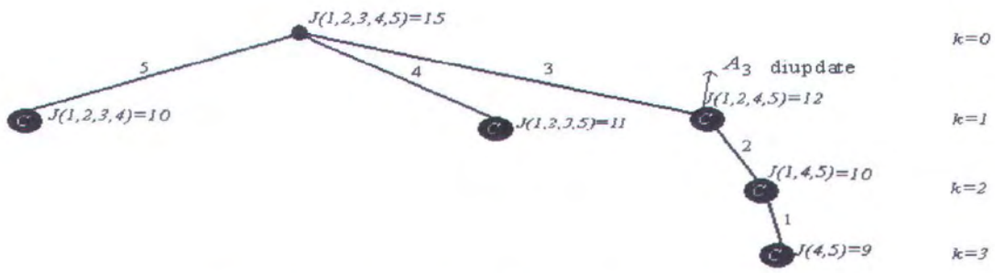
  $Q_{2,1} = \psi_1 = \xi_1$

  $J_{2,1} = v_1 = 9$

  $T_{2,1} = "C"$

  dan didapatkan $Q_2 = \{\xi_1\}$, $J_1 = [9]^T$, dan $T_2 = ["C"]^T$

Dari perhitungan diatas didapat node-node keturunan dari *current* node sebanyak 1 dan *tree*-nya dapat dilihat pada Gambar 3.10 berikut.



Gambar 3.10 Turunan node pada *node tree level* 2

Untuk menghindari duplikasi testing, maka fitur ψ_{j_i} dihilangkan dari himpunan fitur Ψ, yaitu sebagai berikut:

- $\Psi = \Psi \setminus Q_2$ dan $r = r - q_2$

  $\Psi = \phi$ $r = 1 - 1 = 0$

Kemudian dilanjutkan ke **Langkah 2**.

Update A_2 dan S_2

A_2 dan S_2 diupdate sebagai berikut:

$$A_2 = \frac{A_2 . S_2 + J_{0,3} - J\left(\overline{x_1} \setminus \{\xi_2\}\right)}{S_2 + 1}$$

$$A_2 = \frac{0.0 + 12 - 10}{1}$$

$$A_2 = 2$$

dan

$$S_2 = S_2 + 1$$

$$S_2 = 0 + 1 = 1$$

Langkah 2 Memeriksa turunan node paling kanan

Telah diketahui

$k = 2$, $r = 0$, $\overline{x_2} = \{\xi_1, \xi_4, \xi_5\}$, $\Psi = \phi$, $x^* = \sim$

Selanjutnya diperiksa:

- *If* $q_2 = 0$

  Telah diketahui bahwa $q_2 = 1$. Sehingga hal ini tidak memenuhi kondisi $q_2 = 0$.

- If $(T_{2,1} = "P")$ *AND* $J_{2,1} < x^*$

  Telah diketahui bahwa $T_{2,1} = "C"$ dan $J_{2,1} > x^*$, maka hal ini tidak memenuhi kondisi yang diminta.

- If $(T_{2,1} = "C")$ *AND* $J_{2,1} < x^*$

  Telah diketahui bahwa $T_{2,1} = "C"$ dan $J_{2,1} > x^*$, maka hal ini tidak memenuhi kondisi yang diminta.

- Else

  $\overline{X_{2+1}} = \overline{X_2} \setminus \{Q_{2,1}\}$

  $\overline{x_3} = \{\xi_4, \xi_5\}$

  - *If* $k+1 = D-d$

    Telah diketahui bahwa $k+1 = 3$ dan $D-d = 3$, sehingga memenuhi kondisi yang diminta, dan *leaf* telah ditemukan. Kemudian dilanjutkan ke **Langkah 5**.

Langkah 5

Nilai *bound* diupdate sebagai berikut:

$x^* = J_{2,1}$

$x^* = 9$

dan

$$x = \overline{x_{2+1}}$$

$$x = \overline{x_3}$$

$$x = \{\xi_4, \xi_5\}$$

Kemudian dilanjutkan ke **Langkah 2.**

Update A_1 dan S_1

A_1 dan S_1 diupdate sebagai berikut:

$$A_1 = \frac{A_1 . S_1 + J_{1,1} - J(\overline{x_2} \setminus \{\xi_1\})}{S_1 + 1}$$

$$A_1 = \frac{0.0 + 10 - 9}{1}$$

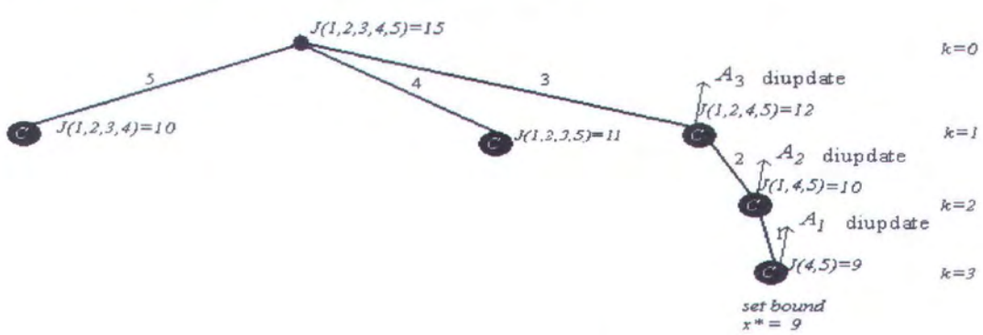
$$A_1 = 1$$

dan

$$S_1 = S_1 + 1$$

$$S_1 = 0 + 1 = 1$$

Dari perhitungan diatas ditemukan *leaf* dan didapat nilai *bound*. Gambar 3.11 berikut adalah *tree* setelah nilai *bound* didapat.



Gambar 3.11 *Leaf* telah ditemukan dan didapatkan nilai *bound*

Langkah 2 Memeriksa turunan node paling kanan

Telah diketahui

$k=2,\ r=0,\ \overline{x_2}=\{\xi_1,\xi_4,\xi_5\},\ \overline{x_3}=\{\xi_4,\xi_5\},\ \Psi=\phi,\ x^*=9$

Selanjutnya diperiksa:

- *If* $q_2=0$

  Telah diketahui bahwa $q_2=1$. Sehingga hal ini tidak memenuhi kondisi $q_2=0$.

- If $(T_{2,1}=\text{"}P\text{"})\ AND\ J_{2,1}<x^*$

  Telah diketahui bahwa $T_{2,1}=\text{"}C\text{"}$ dan $J_{2,1}<x^*$, maka hal ini tidak memenuhi kondisi yang diminta.

- If $(T_{2,1}=\text{"}C\text{"})\ AND\ J_{2,1}<x^*$

  Telah diketahui bahwa $T_{2,1}=\text{"}C\text{"}$ dan $J_{2,1}<x^*$, maka memenuhi kondisi yang diminta dan dilanjutkan ke **Langkah 3.**

Langkah 3 Turunan node mungkin di-*cut-off*

Mengembalikan fitur $Q_{2,1}$ pada himpunan fitur Ψ, yaitu:

- $\Psi=\Psi\cup\{Q_{2,1}\}$ dan $r=r+1$

  $\Psi=\{\xi_1\}$ $r=0+1=1$

- $Q_2=Q_2\setminus\{Q_{2,1}\}$ dan $q_2=q_2-1$

  $Q_2=\phi$ $q_2=1-1=0$.

Kemudian dilanjutkan ke **Langkah 2.**

Langkah 2 Memeriksa turunan node paling kanan

Telah diketahui

$k=2,\ r=1,\ \overline{x_2}=\{\xi_1,\xi_4,\xi_5\},\ \overline{x_3}=\{\xi_4,\xi_5\},\ \Psi=\{\xi_1\},\ x^*=9$

Selanjutnya diperiksa:

- *If* $q_2 = 0$

  Telah diketahui bahwa $q_2 = 0$. Sehingga memenuhi kondisi $q_2 = 0$ dan dilanjutkan ke **Langkah 4**.

Langkah 4 *Backtracking*:

- $k = k - 1$

  $k = 2 - 1$

  $k = 1$

  - If $k = -1$

    Karena $k = 1$, maka hal ini tidak memenuhi kondisi yang diminta.

  - Else

    Mengembalikan fitur $Q_{1,1}$ pada himpunan kandidat. Sehingga :

    $\overline{X_1} = \overline{X_2} \cup \{Q_{1,1}\}$

    $\overline{x_1} = \{\xi_1, \xi_2, \xi_4, \xi_5\}$

    Kemudian dilanjutkan ke **Langkah 3.**

Langkah 3

Mengembalikan fitur $Q_{2,1}$ pada himpunan fitur Ψ, yaitu:

- $\Psi = \Psi \cup \{Q_{1,1}\}$ dan $r = r + 1$

  $\Psi = \{\xi_1, \xi_2\}$ $r = 1 + 1 = 2$

- $Q_1 = Q_1 \setminus \{Q_{1,1}\}$ dan $q_1 = q_1 - 1$

$Q_1 = \phi$ $q_1 = 1-1 = 0$.

Kemudian dilanjutkan ke **Langkah 2.**

Langkah 2 Memeriksa turunan node paling kanan

Telah diketahui

$k = 1$, $r = 2$, $\overline{x_1} = \{\xi_1, \xi_2, \xi_4, \xi_5\}$, $\Psi = \{\xi_1, \xi_2\}$, $x^* = 9$

Selanjutnya diperiksa:

- *If* $q_1 = 0$

  Telah diketahui bahwa $q_1 = 0$. Sehingga memenuhi kondisi $q_1 = 0$ dan dilanjutkan ke **Langkah 4**.

Langkah 4 *Backtracking*:

- $k = k - 1$

  $k = 1 - 1$

  $k = 0$

  - If $k = -1$

    Karena $k = 0$, maka hal ini tidak memenuhi kondisi yang diminta.

  - Else

    Mengembalikan fitur $Q_{0,3}$ pada himpunan kandidat. Sehingga :

    $\overline{X_0} = \overline{X_1} \cup \{Q_{0,3}\}$

    $\overline{x_0} = \{\xi_1, \xi_2, \xi_3, \xi_4, \xi_5\}$

    Kemudian dilanjutkan ke **Langkah 3.**

Langkah 3

Mengembalikan fitur $Q_{0,3}$ pada himpunan fitur Ψ, yaitu:

- $\Psi = \Psi \cup \{Q_{0,3}\}$ dan $r = r + 1$

  $\Psi = \{\xi_1, \xi_2, \xi_3\}$ $r = 2 + 1 = 3$

- $Q_0 = Q_0 \setminus \{Q_{0,3}\}$ dan $q_0 = q_0 - 1$

  $Q_1 = \{\xi_5, \xi_4\}$ $q_0 = 3 - 1 = 2$.

Kemudian dilanjutkan ke **Langkah 2.**

Langkah 2 Memeriksa turunan node paling kanan

Telah diketahui

$k = 0$, $r = 3$, $\overline{x_0} = \{\xi_1, \xi_2, \xi_3, \xi_4, \xi_5\}$, $\Psi = \{\xi_1, \xi_2, \xi_3\}$, $x^* = 9$

Selanjutnya diperiksa:

- *If* $q_0 = 0$

  Telah diketahui bahwa $q_0 = 2$. Sehingga hal ini tidak memenuhi kondisi $q_1 = 0$ dan.

- If $(T_{0,2} = "P")\ AND\ J_{0,2} < x^*$

  Telah diketahui bahwa $T_{0,2} = "C"$ dan $J_{0,2} > x^*$, maka hal ini tidak memenuhi kondisi yang diminta.

- If $(T_{0,2} = "C")\ AND\ J_{0,2} < x^*$

  Telah diketahui bahwa $T_{0,2} = "C"$ dan $J_{0,2} > x^*$, maka hal ini tidak memenuhi kondisi yang diminta.

- Else

  $\overline{X_{0+1}} = \overline{X_0} \setminus \{Q_{0,2}\}$

$\overline{x_1} = \{\xi_1, \xi_2, \xi_3, \xi_5\}$

- *If* $k+1 = D-d$

  Telah diketahui bahwa $k+1=1$ dan $D-d=3$, sehingga hal ini tidak memenuhi kondisi yang diminta.

- *Else*

  $k = k+1$

  $k = 0+1$

  $k = 1$

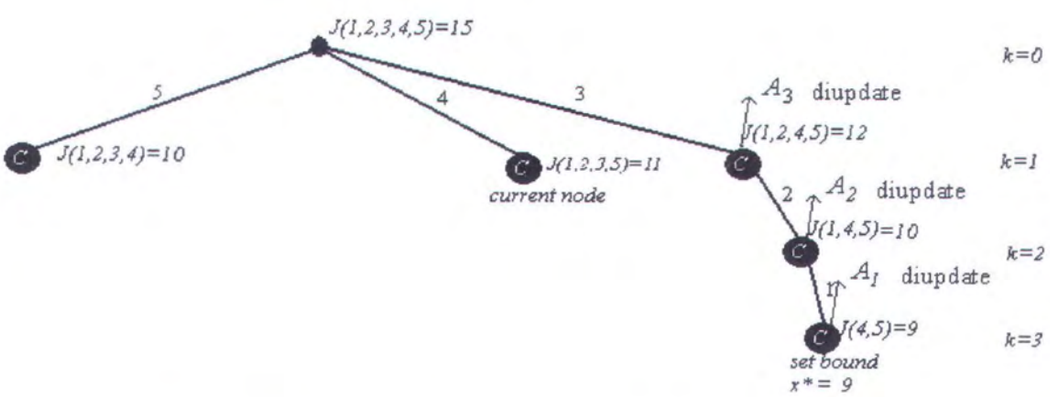
dan dilanjutkan ke **Langkah 1**

Langkah 1 Memilih keturunan dari *current* node untuk membentuk level pohon yang berhubungan

Telah diketahui

$k=1$, $r=3$, $\overline{x_1} = \{\xi_1, \xi_2, \xi_3, \xi_5\}$, $\Psi = \{\xi_1, \xi_2, \xi_3\}$, $x^* = 9$

pada gambar 3.12 dibawah ini menunjukkan posisi *current* node.



Gambar 3.12 ***Node*** **dengan** $J(1,2,3,5)$ **sebagai** ***current node***

maka:

- $q_1 = 3-(5-2-1-1)$

$q_1 = 2$

- Untuk j = 1 sampai r sehingga nilai j = 1, 2, 3

$\psi_j \in \Psi$

untuk j = 1, didapatkan:

$\psi_1 = \xi_3$

- *If* $(k+1 < D-d)$ *and* $S_3 > \delta$

  Telah diketahui sebelumnya bahwa $k+1=2$ dan $D-d=3$ maka hal ini memenuhi kondisi $(k+1<D-d)$. Dan karena $S_3 = 1$ dan $\delta = 0$ maka memenuhi kondisi $S_3 > \delta$. Sehingga prediksi diijinkan dan didapat:

  $v_1 = J_{0,2} - A_3$

  $v_1 = 11-3$

  $v_1 = 8$

untuk j = 2, didapatkan:

$\psi_2 = \xi_2$

- *If* $(k+1 < D-d)$ *and* $S_2 > \delta$

  Telah diketahui sebelumnya bahwa $k+1=2$ dan $D-d=3$ maka hal ini memenuhi kondisi $(k+1<D-d)$. Dan karena $S_2 = 1$ dan $\delta = 0$ maka memenuhi kondisi $S_2 > \delta$. Sehingga prediksi diijinkan dan didapat:

  $v_2 = J_{0,2} - A_2$

  $v_2 = 11-2$

$v_1 = 9$

untuk j = 3, didapatkan:

$\psi_3 = \xi_1$

- *If* $(k+1 < D-d)$ *and* $S_1 > \delta$

  Telah diketahui sebelumnya bahwa $k+1=2$ dan $D-d=3$ maka hal ini memenuhi kondisi $(k+1<D-d)$. Dan karena $S_1 = 1$ dan $\delta = 0$ maka memenuhi kondisi $S_1 > \delta$. Sehingga prediksi diijinkan dan didapat:

  $v_1 = J_{0,2} - A_1$

  $v_{14} = 11-1$

  $v_1 = 10$

sehingga $v_1 \le v_2 \le v_3$

- untuk i = 1,...,q_1 sehingga nilai i = 1, 2 dan didapatkan:

  untuk i = 1

  $Q_{1,1} = \psi_1 = \xi_3$

  $J_{1,1} = 8$

  $T_{1,1} = "P"$

  untuk i = 2

  $Q_{1,2} = \psi_2 = \xi_2$

  $J_{1,2} = 9$

  $T_{1,2} = "P"$

  dan didapatkan $Q_1 = \{\xi_3, \xi_2\}$.

- Untuk menghindari duplikasi testing, maka fitur ψ_{j_1} dihilangkan dari himpunan fitur Ψ, yaitu sebagai berikut:

  - $\Psi = \Psi \setminus Q_1$ dan $r = r - q_1$

    $\Psi = \{\xi_1\}$ $r = 3 - 2 = 1$

  Kemudian dilanjutkan ke **Langkah 2**.

Update A_4 dan S_4

A_4 dan S_4 diupdate sebagai berikut:

$$A_4 = \frac{A_4 . S_4 + J_{-1,y_j} - J(\overline{x_0} \setminus \{\xi_4\})}{S_4 + 1}$$

$$A_4 = \frac{0.0 + 15 - 11}{1}$$

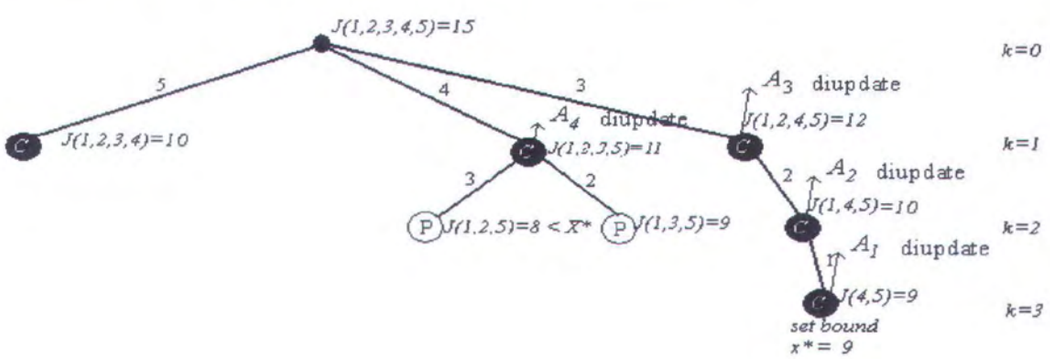
$$A_4 = 4$$

dan

$$S_4 = S_4 + 1$$

$$S_4 = 0 + 1 = 1$$

Dari perhitungan diatas didapat node-node keturunan dari *current* node sebanyak 2 dan *tree*-nya dapat dilihat pada Gambar 3.13 berikut.



Gambar 3.13 Turunan *node* pada *node tree* level 1 dari *current node*

Langkah 2 Memeriksa turunan node paling kanan

Telah diketahui

$k=1,\ r=1,\ \overline{x_1}=\{\xi_1,\xi_2,\xi_3,\xi_5\},\ \Psi=\{\xi_1\},\ x^*=9$

Selanjutnya diperiksa:

- *If* $q_1=0$

  Telah diketahui bahwa $q_1=2$. Sehingga hal ini tidak memenuhi kondisi $q_1=0$ dan.

- If $(T_{1,2}="P")\ AND\ J_{1,2}<x^*$

  Telah diketahui bahwa $T_{1,2}="P"$ dan $J_{1,2}=x^*$, maka hal ini tidak memenuhi kondisi yang diminta.

- If $(T_{1,2}="C")\ AND\ J_{1,2}<x^*$

  Telah diketahui bahwa $T_{1,2}="P"$ dan $J_{1,2}=x^*$, maka hal ini tidak memenuhi kondisi yang diminta.

- Else

  $\overline{X_{1+1}}=\overline{X_1}\setminus\{Q_{1,2}\}$

  $\overline{x_2}=\{\xi_1,\xi_3,\xi_5\}$

  - *If* $k+1=D-d$

    Telah diketahui bahwa $k+1=2$ dan $D-d=3$, sehingga hal ini tidak memenuhi kondisi yang diminta.*Else*

    $k=k+1$

    $k=1+1$

$$k = 2$$

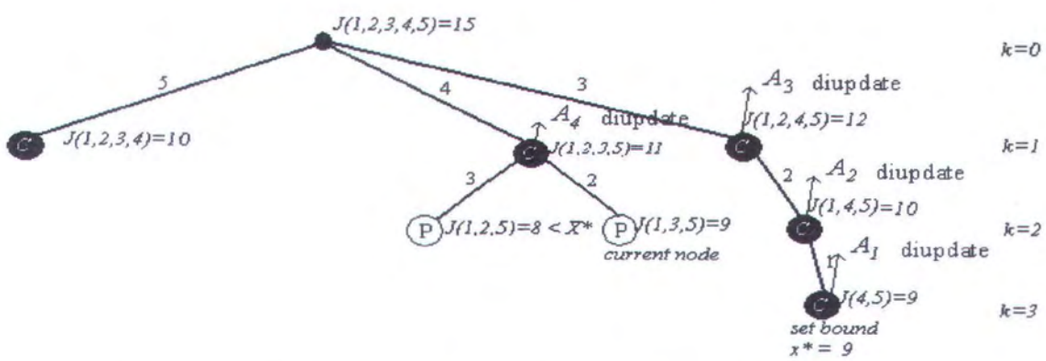
dan dilanjutkan ke **Langkah 1**

Langkah 1 Memilih keturunan dari *current* node untuk membentuk level pohon yang berhubungan

Telah diketahui

$$k = 2,\ r = 1,\ \overline{x_2} = \{\xi_1, \xi_3, \xi_5\},\ \Psi = \{\xi_1\},\ x^* = 9$$

pada gambar 3.14 dibawah ini menunjukkan posisi *current* node.



Gambar 3.14 Node dengan $J(1,3,5)$ sebagai *current* node

maka:

- $q_2 = 1-(5-2-2-1)$

  $q_2 = 1$

- Untuk j = 1 sampai r sehingga nilai j = 1

  $\psi_j \in \Psi$

  untuk j = 1, didapatkan:

  $\psi_1 = \xi_1$

  - *If* $(k+1 < D-d)$ *and* $S_1 > \delta$

Telah diketahui sebelumnya bahwa $k+1=3$ dan $D-d=3$ maka hal ini tidak memenuhi kondisi $(k+1<D-d)$. Dan karena $S_1=1$ dan $\delta=0$ maka memenuhi kondisi $S_1>\delta$. Sehingga prediksi tidak diijinkan

- *Else*

  Maka nilai kriterianya harus dihitung ("C"), yaitu:

  $v_1 = J(\overline{x_2} \setminus \{\xi_1\})$

  $v_1 = J(3,5)$

  $v_1 = 8$

- untuk i = 1,...,q_2 sehingga nilai i = 1, dan didapatkan:

  untuk i = 1

  $Q_{2,1} = \psi_1 = \xi_1$

  $J_{2,1} = 8$

  $T_{2,1} = "C"$

  dan didapatkan $Q_2 = \{\xi_1\}$.

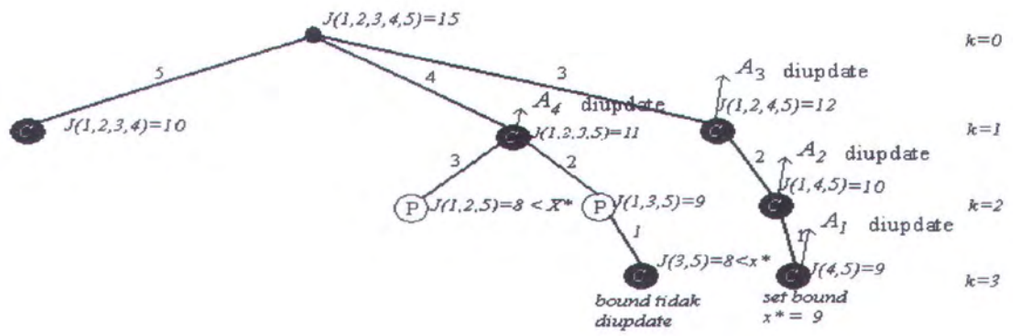
- Untuk menghindari duplikasi testing, maka fitur ψ_{j_i} dihilangkan dari himpunan fitur Ψ, yaitu sebagai berikut:

  - $\Psi = \Psi \setminus Q_2$ dan $r = r - q_2$

    $\Psi = \phi$ $r = 1-1 = 0$

  Kemudian dilanjutkan ke **Langkah 2**.

Dari perhitungan diatas didapat node-node keturunan dari *current* node sebanyak 1 dan *tree*-nya dapat dilihat pada Gambar 3.15 berikut.



Gambar 3.15 Turunan *node* pada *node tree* level 3 dari *current node*

Langkah 2 Memeriksa turunan node paling kanan

Telah diketahui

$$k=2,\ r=0,\ \overline{x_2}=\{\xi_1,\xi_3,\xi_5\},\ \Psi=\phi,\ x^*=9$$

Selanjutnya diperiksa:

- *If* $q_2=0$

  Telah diketahui bahwa $q_2=1$. Sehingga hal ini tidak memenuhi kondisi $q_2=0$.

- If $(T_{2,1}="P")\ AND\ J_{2,1}<x^*$

  Telah diketahui bahwa $T_{2,1}="C"$ dan $J_{2,1}<x^*$, maka hal ini tidak memenuhi kondisi yang diminta.

- If $(T_{2,1}="C")\ AND\ J_{2,1}<x^*$

  Telah diketahui bahwa $T_{2,1}="C"$ dan $J_{2,1}<x^*$, maka hal ini memenuhi kondisi yang diminta dan dilanjutkan ke **Langkah 3**.

Langkah 3 Turunan node mungkin di-*cut-off*

Mengembalikan fitur $Q_{2,1}$ pada himpunan fitur Ψ, yaitu:

- $\Psi=\Psi\cup\{Q_{2,1}\}$ dan $r=r+1$

$\Psi = \{\xi_1\}$ $r = 0+1 = 1$

- $Q_2 = Q_2 \setminus \{Q_{2,1}\}$ dan $q_2 = q_2 - 1$

  $Q_2 = \phi$ $q_2 = 1-1 = 0$.

Kemudian dilanjutkan ke **Langkah 2.**

Langkah 2 Memeriksa turunan node paling kanan

Telah diketahui

$k = 2$, $r = 1$, $\overline{x_2} = \{\xi_1, \xi_3, \xi_5\}$, $\Psi = \{\xi_1\}$, $x^* = 9$

Selanjutnya diperiksa:

- *If* $q_2 = 0$

  Telah diketahui bahwa $q_2 = 0$. Sehingga memenuhi kondisi $q_2 = 0$ dan dilanjutkan ke **Langkah 4**.

Langkah 4 *Backtracking*:

- $k = k - 1$

  $k = 2 - 1$

  $k = 1$

  - If $k = -1$

    Karena $k = 1$, maka hal ini tidak memenuhi kondisi yang diminta.

  - Else

    Mengembalikan fitur $Q_{1,1}$ pada himpunan kandidat. Sehingga :

    $\overline{X_1} = \overline{X_2} \cup \{Q_{1,2}\}$

$$\overline{x_1} = \{\xi_1, \xi_2, \xi_3, \xi_5\}$$

Kemudian dilanjutkan ke **Langkah 3.**

Langkah 3

Mengembalikan fitur $Q_{1,1}$ pada himpunan fitur Ψ, yaitu:

- $\Psi = \Psi \cup \{Q_{1,2}\}$ dan $r = r + 1$

  $\Psi = \{\xi_1, \xi_2\}$ $r = 1 + 1 = 2$

- $Q_1 = Q_1 \setminus \{Q_{1,2}\}$ dan $q_1 = q_1 - 1$

  $Q_1 = \{\xi_3\}$ $q_1 = 2 - 1 = 1$.

Kemudian dilanjutkan ke **Langkah 2.**

Langkah 2 Memeriksa turunan node paling kanan

Telah diketahui

$k = 1$, $r = 2$, $\overline{x_1} = \{\xi_1, \xi_2, \xi_3, \xi_5\}$, $\Psi = \{\xi_1, \xi_2\}$, $x^* = 9$

Selanjutnya diperiksa:

- *If* $q_1 = 0$

  Telah diketahui bahwa $q_1 = 1$. Sehingga hal ini tidak memenuhi kondisi $q_1 = 0$.

- If $(T_{1,1} = "P")$ *AND* $J_{1,1} < x^*$

  Telah diketahui bahwa $T_{1,1} = "P"$ dan $J_{1,1} > x^*$, maka hal ini memenuhi kondisi yang diminta dan nilai kriteria sebenarnya harus dihitung, sehingga didapatkan:

  $J_{1,1} = J(\overline{x_1} \setminus \{Q_{1,1}\})$

  $J_{1,1} = J(1,2,5)$

$J_{1,1} = 8$

dan

$T_{1,1} =$"C"

- If $(T_{1,1} =$"C"$)\ AND\ J_{1,1} < x^*$

  Telah diketahui bahwa $T_{1,1} =$"C" dan $J_{1,1} < x^*$, maka hal ini memenuhi kondisi yang diminta dan dilanjutkan ke **Langkah 3**.

Langkah 3

Mengembalikan fitur $Q_{1,1}$ pada himpunan fitur Ψ, yaitu:

- $\Psi = \Psi \cup \{Q_{1,1}\}$ dan $r = r + 1$

  $\Psi = \{\xi_1, \xi_2, \xi_3\}$ $r = 2 + 1 = 3$

- $Q_1 = Q_1 \setminus \{Q_{1,1}\}$ dan $q_1 = q_1 - 1$

  $Q_1 = \phi$ $q_1 = 1 - 1 = 0$.

Kemudian dilanjutkan ke **Langkah 2.**

Langkah 2 Memeriksa turunan node paling kanan

Telah diketahui

$k = 1$, $r = 3$, $\overline{x_1} = \{\xi_1, \xi_2, \xi_3, \xi_5\}$, $\Psi = \{\xi_1, \xi_2, \xi_3\}$, $x^* = 9$

Selanjutnya diperiksa:

- *If* $q_1 = 0$

  Telah diketahui bahwa $q_1 = 0$. Sehingga memenuhi kondisi $q_1 = 0$ dan dilanjutkan ke **Langkah 4**.

Langkah 4 *Backtracking*:

- $k = k - 1$

$k = 1 - 1$

$k = 0$

- If $k = -1$

  Karena $k = 0$, maka hal ini tidak memenuhi kondisi yang diminta.

- Else

  Mengembalikan fitur $Q_{0,2}$ pada himpunan kandidat. Sehingga :

  $\overline{X_0} = \overline{X_1} \cup \{Q_{0,2}\}$

  $\overline{x_0} = \{\xi_1, \xi_2, \xi_3, \xi_4, \xi_5\}$

  Kemudian dilanjutkan ke **Langkah 3.**

Langkah 3

Mengembalikan fitur $Q_{0,2}$ pada himpunan fitur Ψ, yaitu:

- $\Psi = \Psi \cup \{Q_{0,2}\}$ dan $r = r + 1$

  $\Psi = \{\xi_1, \xi_2, \xi_3, \xi_4\}$ $r = 3 + 1 = 4$

- $Q_0 = Q_0 \setminus \{Q_{0,2}\}$ dan $q_0 = q_0 - 1$

  $Q_0 = \{\xi_5\}$ $q_0 = 2 - 1 = 1$.

Kemudian dilanjutkan ke **Langkah 2.**

Langkah 2 Memeriksa turunan node paling kanan

Telah diketahui :

$k = 0$, $q_0 = 1$, $r = 4$, $\Psi = \{\xi_1, \xi_2, \xi_3, \xi_4\}$, $\overline{x_0} = \{\xi_1, \xi_2, \xi_3, \xi_4, \xi_5\}$, $x^* = 9$, $Q_0 = \{\xi_5\}$

Selanjutnya diperiksa:

- *If* $q_0 = 0$

  Telah diketahui bahwa $q_0 = 1$. Sehingga hal ini tidak memenuhi kondisi $q_0 = 0$.

- If $(T_{0,1} = "P")$ *AND* $J_{0,1} < x^*$

  Telah diketahui bahwa $T_{0,1} = "C"$ dan $J_{0,1} > x^*$, maka hal ini tidak memenuhi kondisi yang diminta.

- If $(T_{0,1} = "C")$ *AND* $J_{0,1} < x^*$

  Telah diketahui bahwa $T_{0,1} = "C"$ dan $J_{0,1} > x^*$, maka hal ini tidak memenuhi kondisi yang diminta.

- Else

  $\overline{X_{0+1}} = \overline{X_0} \setminus \{Q_{0,1}\}$

  $\overline{x_1} = \{\xi_1, \xi_2, \xi_3, \xi_4\}$

  - *If* $k+1 = D-d$

    Telah diketahui bahwa $k+1 = 1$ dan $D-d = 3$, sehingga hal ini tidak memenuhi kondisi yang diminta.

  - *Else*

    $k = k+1$

    $k = 0+1$

    $k = 1$

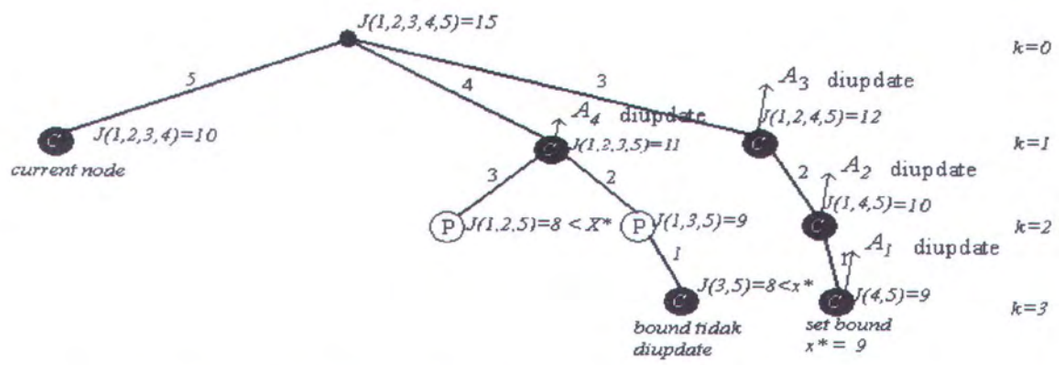
    dan dilanjutkan ke **Langkah 1**

Langkah 1 Memilih keturunan dari *current* node untuk membentuk level pohon yang berhubungan

Telah diketahui

$$k=1,\ r=4,\ \overline{x_1}=\{\xi_1,\xi_2,\xi_3,\xi_4\},\ \Psi=\{\xi_1,\xi_2,\xi_3,\xi_4\},\ x^*=9$$

pada gambar 3.16 dibawah ini menunjukkan posisi *current* node.



Gambar 3.16 *Node* **dengan** $J(1,2,3,4)$ **sebagai** ***current node***

maka:

- $q_1 = 4-(5-2-1-1)$

  $q_1 = 3$

- Untuk j = 1 sampai r sehingga nilai j = 1, 2, 3, 4

  $\psi_j \in \Psi$

  untuk j = 1, didapatkan:

  $\psi_1 = \xi_4$

  - *If* $(k+1<D-d)$ *and* $S_4 > \delta$

    Telah diketahui sebelumnya bahwa $k+1=2$ dan $D-d=3$

    maka hal ini memenuhi kondisi $(k+1<D-d)$, dan karena

$S_4 = 1$ dan $\delta = 0$ maka hal ini memenuhi kondisi $S_2 > \delta$.

Sehingga prediksi diijinkan dan didapat:

$v_1 = J_{0,1} - A_4$

$v_1 = 10 - 4$

$v_1 = 6$

untuk j = 2, didapatkan:

$\psi_2 = \xi_3$

- *If* $(k+1 < D-d)$ *and* $S_3 > \delta$

  Telah diketahui sebelumnya bahwa $k+1 = 2$ dan $D-d = 3$ maka hal ini memenuhi kondisi $(k+1 < D-d)$, dan karena $S_3 = 1$ dan $\delta = 0$ maka hal ini memenuhi kondisi $S_3 > \delta$.

  Sehingga prediksi diijinkan dan didapat:

  $v_2 = J_{0,1} - A_3$

  $v_2 = 10 - 3$

  $v_2 = 7$

untuk j = 3, didapatkan:

$\psi_3 = \xi_2$

- *If* $(k+1 < D-d)$ *and* $S_2 > \delta$

  Telah diketahui sebelumnya bahwa $k+1 = 2$ dan $D-d = 3$ maka hal ini memenuhi kondisi $(k+1 < D-d)$, dan karena $S_2 = 1$ dan $\delta = 0$ maka hal ini memenuhi kondisi $S_2 > \delta$.

  Sehingga prediksi diijinkan dan didapat:

$$v_3 = J_{0,1} - A_2$$

$$v_3 = 10 - 2$$

$$v_3 = 8$$

untuk j = 4, didapatkan:

$$\psi_4 = \xi_1$$

- *If* $(k+1 < D-d)$ *and* $S_1 > \delta$

  Telah diketahui sebelumnya bahwa $k+1=2$ dan $D-d=3$ maka hal ini memenuhi kondisi $(k+1 < D-d)$, dan karena $S_1 = 1$ dan $\delta = 0$ maka hal ini memenuhi kondisi $S_1 > \delta$. Sehingga prediksi diijinkan dan didapat:

  $$v_4 = J_{0,1} - A_1$$

  $$v_4 = 10 - 1$$

  $$v_4 = 9$$

sehingga $v_1 \le v_2 \le v_3 \le v_4$

- untuk i = 1,...,q_1 sehingga nilai i = 1, 2, 3 dan didapatkan:

  untuk i = 1

  $$Q_{1,1} = \psi_1 = \xi_4$$

  $$J_{1,1} = v_1 = 6$$

  $$T_{1,1} = "P"$$

  untuk i = 2

  $$Q_{1,2} = \psi_2 = \xi_3$$

  $$J_{1,2} = v_2 = 7$$

$T_{1,2}$ ="P"

untuk i = 3

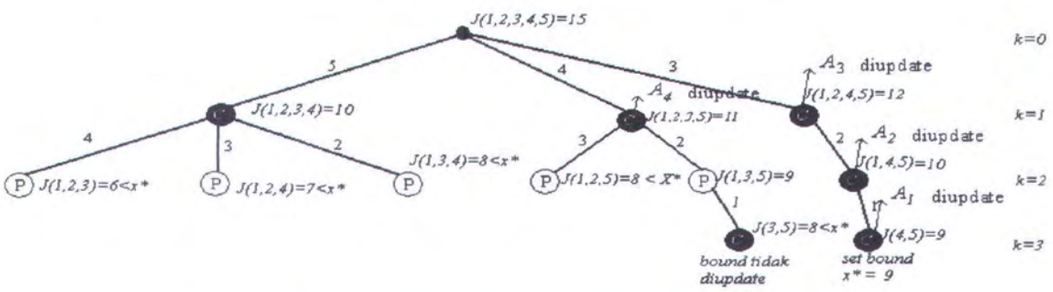
$Q_{1,3} = \psi_3 = \xi_2$

$J_{1,3} = v_3 = 8$

$T_{1,3}$ ="P"

dan didapatkan $Q_1 = \{\xi_4, \xi_3, \xi_2\}$, $J_1 = [6 \quad 7 \quad 8]^T$, dan $T_1 = [\text{"}P\text{"} \quad \text{"}P\text{"} \quad \text{"}P\text{"}]^T$

Dari perhitungan diatas didapat node-node keturunan dari *current* node sebanyak 3 dan *tree*-nya dapat dilihat pada Gambar 3.17 berikut.



Gambar 3.17 Turunan *node* pada *node tree level* 1

Untuk menghindari duplikasi testing, maka fitur ψ_{j_t} dihilangkan dari himpunan fitur Ψ, yaitu sebagai berikut:

- $\Psi = \Psi \setminus Q_1$ dan $r = r - q_1$

  $\Psi = \{\xi_1\}$ $r = 4 - 3 = 1$

Kemudian dilanjutkan ke **Langkah 2**.

Update A_5 dan S_5

A_5 dan S_5 diupdate sebagai berikut:

$$A_5 = \frac{A_5 . S_5 + J_{-1,y_j} - J(\overline{x_1} \setminus \{\xi_5\})}{S_5 + 1}$$

$$A_5 = \frac{0.0+15-10}{1}$$

$$A_5 = 5$$

dan

$$S_5 = S_5 + 1$$

$$S_5 = 0+1 = 1$$

Langkah 2 Memeriksa turunan node paling kanan

Telah diketahui :

$$k = 1,\ r = 1,\ \overline{x_1} = \{\xi_1, \xi_2, \xi_3, \xi_4\},\ \Psi = \{\xi_1\},\ x^* = 9$$

Selanjutnya diperiksa:

- *If* $q_1 = 0$

  Telah diketahui bahwa $q_1 = 3$. Sehingga hal ini tidak memenuhi kondisi $q_1 = 0$.

- If $(T_{1,3} = "P")\ AND\ J_{1,3} < x^*$

  Telah diketahui bahwa $T_{1,3} = "P"$ dan $J_{1,3} < x^*$, maka hal ini memenuhi kondisi yang diminta dan nilai kriteria sebenarnya harus dihitung, sehingga didapatkan:

  $$J_{1,3} = J(\overline{x_1} \setminus \{Q_{1,3}\})$$

  $$J_{1,3} = J(1,3,4)$$

  $$J_{1,3} = 8$$

  dan

  $$T_{1,3} = "C"$$

- If $(T_{1,3} = "C")\ AND\ J_{1,3} < x^*$

  Telah diketahui bahwa $T_{1,3} = "C"$ dan $J_{1,3} < x^*$, maka hal ini memenuhi kondisi yang diminta dan dilanjutkan ke **Langkah 3**.

Langkah 3

Mengembalikan fitur $Q_{1,3}$ pada himpunan fitur Ψ, yaitu:

- $\Psi = \Psi \cup \{Q_{1,3}\}$ dan $r = r + 1$

  $\Psi = \{\xi_1, \xi_2\}$ $r = 1 + 1 = 2$

- $Q_1 = Q_1 \setminus \{Q_{1,3}\}$ dan $q_1 = q_1 - 1$

  $Q_1 = \{\xi_4, \xi_3\}$ $q_1 = 3 - 1 = 2$.

Kemudian dilanjutkan ke **Langkah 2.**

Langkah 2 Memeriksa turunan node paling kanan

Telah diketahui :

$k = 1$, $r = 2$, $\overline{x_1} = \{\xi_1, \xi_2, \xi_3, \xi_4\}$, $\Psi = \{\xi_1, \xi_2\}$, $x^* = 9$

Selanjutnya diperiksa:

- *If* $q_1 = 0$

  Telah diketahui bahwa $q_1 = 2$. Sehingga hal ini tidak memenuhi kondisi $q_1 = 0$.

- If $(T_{1,2} = "P")\ AND\ J_{1,2} < x^*$

  Telah diketahui bahwa $T_{1,2} = "P"$ dan $J_{1,2} < x^*$, maka hal ini memenuhi kondisi yang diminta dan nilai kriteria sebenarnya harus dihitung, sehingga didapatkan:

  $$J_{1,2} = J(\overline{x_1} \setminus \{Q_{1,2}\})$$

$J_{1,2} = J(1,2,4)$

$J_{1,2} = 7$

dan

$T_{1,2} = "C"$

- If $(T_{1,2} = "C")\ AND\ J_{1,2} < x^*$

  Telah diketahui bahwa $T_{1,2} = "C"$ dan $J_{1,2} < x^*$, maka hal ini memenuhi kondisi yang diminta dan dilanjutkan ke **Langkah 3**.

Langkah 3

Mengembalikan fitur $Q_{1,2}$ pada himpunan fitur Ψ, yaitu:

- $\Psi = \Psi \cup \{Q_{1,2}\}$ dan $r = r + 1$

  $\Psi = \{\xi_1, \xi_2, \xi_3\}$ $r = 2 + 1 = 3$

- $Q_1 = Q_1 \setminus \{Q_{1,2}\}$ dan $q_1 = q_1 - 1$

  $Q_1 = \{\xi_4\}$ $q_1 = 2 - 1 = 1$.

Kemudian dilanjutkan ke **Langkah 2.**

Langkah 2 Memeriksa turunan node paling kanan

Telah diketahui :

$k = 1$, $r = 3$, $\overline{x_1} = \{\xi_1, \xi_2, \xi_3, \xi_4\}$, $\Psi = \{\xi_1, \xi_2, \xi_3\}$, $x^* = 9$

Selanjutnya diperiksa:

- *If* $q_1 = 0$

  Telah diketahui bahwa $q_1 = 1$. Sehingga hal ini tidak memenuhi kondisi $q_1 = 0$.

- If $(T_{1,1} = "P")\ AND\ J_{1,1} < x^*$

Telah diketahui bahwa $T_{1,1}$ ="P" dan $J_{1,1} < x^*$, maka hal ini memenuhi kondisi yang diminta dan nilai kriteria sebenarnya harus dihitung, sehingga didapatkan:

$$J_{1,1} = J(\overline{x_1} \setminus \{Q_{1,1}\})$$

$$J_{1,1} = J(1,2,3)$$

$$J_{1,1} = 6$$

dan

$$T_{1,1} = "C"$$

- If $(T_{1,1} = "C")\ AND\ J_{1,1} < x^*$

  Telah diketahui bahwa $T_{1,1}$ ="C" dan $J_{1,1} < x^*$, maka hal ini memenuhi kondisi yang diminta dan dilanjutkan ke **Langkah 3**.

Langkah 3

Mengembalikan fitur $Q_{1,1}$ pada himpunan fitur Ψ, yaitu:

- $\Psi = \Psi \cup \{Q_{1,1}\}$ dan $r = r + 1$

  $\Psi = \{\xi_1, \xi_2, \xi_3, \xi_4\}$ $r = 3 + 1 = 4$

- $Q_1 = Q_1 \setminus \{Q_{1,1}\}$ dan $q_1 = q_1 - 1$

  $Q_1 = \phi$ $q_1 = 1 - 1 = 0$.

Kemudian dilanjutkan ke **Langkah 2.**

Langkah 2 Memeriksa turunan node paling kanan

Telah diketahui

$$k = 1,\ r = 4,\ \overline{x_1} = \{\xi_1, \xi_2, \xi_3, \xi_4\},\ \Psi = \{\xi_1, \xi_2, \xi_3, \xi_4\},\ x^* = 9$$

Selanjutnya diperiksa:

- *If* $q_1 = 0$

  Telah diketahui bahwa $q_1 = 0$. Sehingga memenuhi kondisi $q_1 = 0$ dan dilanjutkan ke **Langkah 4**.

Langkah 4 *Backtracking*:

- $k = k - 1$

  $k = 1 - 1$

  $k = 0$

  - If $k = -1$

    Karena $k = 0$, maka hal ini tidak memenuhi kondisi yang diminta.

  - Else

    Mengembalikan fitur $Q_{0,1}$ pada himpunan kandidat. Sehingga :

    $\overline{X_0} = \overline{X_1} \cup \{Q_{0,1}\}$

    $\overline{x_0} = \{\xi_1, \xi_2, \xi_3, \xi_4, \xi_5\}$

    Kemudian dilanjutkan ke **Langkah 3.**

Langkah 3

Mengembalikan fitur $Q_{0,1}$ pada himpunan fitur Ψ, yaitu:

- $\Psi = \Psi \cup \{Q_{0,1}\}$ dan $r = r + 1$

  $\Psi = \{\xi_1, \xi_2, \xi_3, \xi_4, \xi_5\}$ $r = 4 + 1 = 5$

- $Q_0 = Q_0 \setminus \{Q_{0,1}\}$ dan $q_0 = q_0 - 1$

  $Q_0 = \phi$ $q_0 = 1 - 1 = 0$.

Kemudian dilanjutkan ke **Langkah 2.**

Langkah 2 Memeriksa turunan node paling kanan

Telah diketahui

$k = 0$, $r = 4$, $\overline{x_1} = \{\xi_1, \xi_2, \xi_3, \xi_4\}$, $\Psi = \{\xi_1, \xi_2, \xi_3, \xi_4\}$, $x^* = 9$

Selanjutnya diperiksa:

- *If* $q_1 = 0$

  Telah diketahui bahwa $q_1 = 0$. Sehingga memenuhi kondisi $q_1 = 0$ dan dilanjutkan ke **Langkah 4**.

Langkah 4 *Backtracking*:

- $k = k - 1$

  $k = 0 - 1$

  $k = -1$

  - If $k = -1$

    Karena $k = -1$, maka hal ini memenuhi kondisi yang diminta dan algoritma selesai.

Hasil penelusuran *tree* secara keseluruhan menghasilkan jumlah evaluasi kriteria pada algoritma ini sebanyak 10 dengan nilai *bound* yang didapat adalah 9 dan fitur yang dipilih adalah fitur 4 dan fitur 5.

Perbandingan antara algoritma BBB dan algoritma IBB secara sederhana adalah sebagai berikut:

a. Pada BBPP prediksi dilakukan untuk mengurutkan node. untuk memilih node turunan digunakan nilai kriteria sebenarnya. Ketika mengurutkan node, hanya nilai prediksi yang digunakan.

b. Pada FBB prediksi dilakukan untuk mengurutkan node dan untuk memilih node turunan. Ketika mengurutkan node, perhitungan nilai prediksi dan nilai kriteria sebenarnya sama-sama digunakan.

c. FBB lebih banyak mengurangi jumlah penghitungan evaluasi kriteria.

d. FBB lebih *representatif* daripada algoritma BBB, IBB dan BBPP.

3.2.3 Perbandingan Evaluasi Fungsi Kriteria

Setelah dijelaskan keempat algoritma *Branch & Bound* dengan menggunakan satu contoh permasalahan yang menggunakan fungsi kriteria yang sama, maka dapat diketahui evaluasi kriteria dari masing-masing algoritma *Branch & Bound* seperti ditunjukkan pada tabel 3.1 berikut:

Tabel 3.1 Tabel dari evaluasi kriteria pada algoritma *Branch & Bound*

Algoritma *Branch & Bound*	**Jumlah Evaluasi Kriteria**
Basic Branch and Bound	19
Improved Branch and Bound	14
Branch and Bound Partial Prediction	13
Fast Branch and Bound	10

BAB IV

DESAIN PERANGKAT LUNAK

Pada bab ini dijelaskan mengenai desain perangkat lunak algoritma *Branch and Bound* yang digunakan pada pemilihan fitur. Pembahasan ini meliputi lingkungan desain perangkat lunak, masukan dan luaran, desain proses serta antarmuka aplikasi. Desain perangkat lunak ini menggunakan notasi *Flowchart*. *Flowchart* dipergunakan untuk menjelaskan alur proses yang terjadi.

4.1 LINGKUNGAN DESAIN

Pada bagian ini akan dijelaskan mengenai lingkungan pendesainan aplikasi yang meliputi perangkat lunak dan perangkat keras yang digunakan. Spesifikasi perangkat lunak dan perangkat keras yang digunakan dalam pendesainan pemilihan fitur menggunakan algoritma *Branch and Bound* ini dapat dilihat pada tabel 4.1.

Tabel 4.1. Lingkungan Pendesainan Aplikasi

Perangkat Keras	Prosesor : Intel Pentium IV 2.80GHz Memori : 1 GB
Perangkat Lunak	Sistem Operasi : Windows Perangkat Lunak Pendesain : Microsoft Visio 2002

4.2 MASUKAN DAN LUARAN

Masukkan dari sistem ini berupa jumlah elemen himpunan fitur yang akan dipilih dan jumlah fitur yang diinginkan. Luaran dari sistem ini berupa jumlah banyaknya evaluasi fungsi kriteria. Banyaknya evaluasi fungsi kriteria untuk tiap algoritma ditampilkan dalam bentuk grafik kartesian. Grafik evaluasi fungsi kriteria tiap algoritma berada dalam satu grafik kartesian, sehingga bisa dilihat langsung perbedaannya.

4.3 DESAIN ALGORITMA *BRANCH AND BOUND* DASAR

Secara umum pada algoritma *Branch and* Bound dasar ini terdapat 6 langkah, yaitu :

a. Langkah pertama yang disebut dengan inisialisasi.

   Pada langkah ini variabel-variabel yang harus diinisialisasikan lebih dulu yaitu variabel *bound* (x^*), *level* dan z_0. Variabel *bound* diinisialisasikan dengan nol (0), variabel *level* (k) diinisialisasikan dengan 1 dan variabel z_0 diinisialisasikan dengan nol (0).

b. Langkah kedua yang disebut dengan *generate successor* (inisialisasi *LIST(k)*).

   Pada langkah ini didapatkan fitur-fitur yang bisa dijadikan sebagai fitur *candidate*. Fitur *candidate* ini disimpan pada suatu variabel $LIST(k)$. Fitur *candidate* ini merupakan fitur-fitur yang berpotensi untuk dihilangkan.

c. Langkah ketiga yang disebut dengan pemilihan *node* baru (*select new node*).

   Pada langkah ini satu fitur akan dipilih dari fitur *candidate* untuk dijadikan sebagai *node* pada *level tree* yang berhubungan. Setelah satu fitur pada fitur *candidate* dijadikan *node* pada *tree*, maka fitur tersebut akan dihilangkan dari fitur *candidate*.

d. Langkah keempat yang disebut dengan pemeriksaan nilai *bound* (*check bound*).

   Pada langkah ini setiap nilai kriteria pada *level tree* dihitung dan diperiksa untuk dibandingkan dengan nilai *bound* saat itu, apakah lebih kecil atau tidak. Bila nilai kriteria lebih kecil dari nilai *bound* saat itu, maka akan dilanjutkan pada langkah kelima yang menunjukkan bahwa node tersebut mengalami *cut off*.

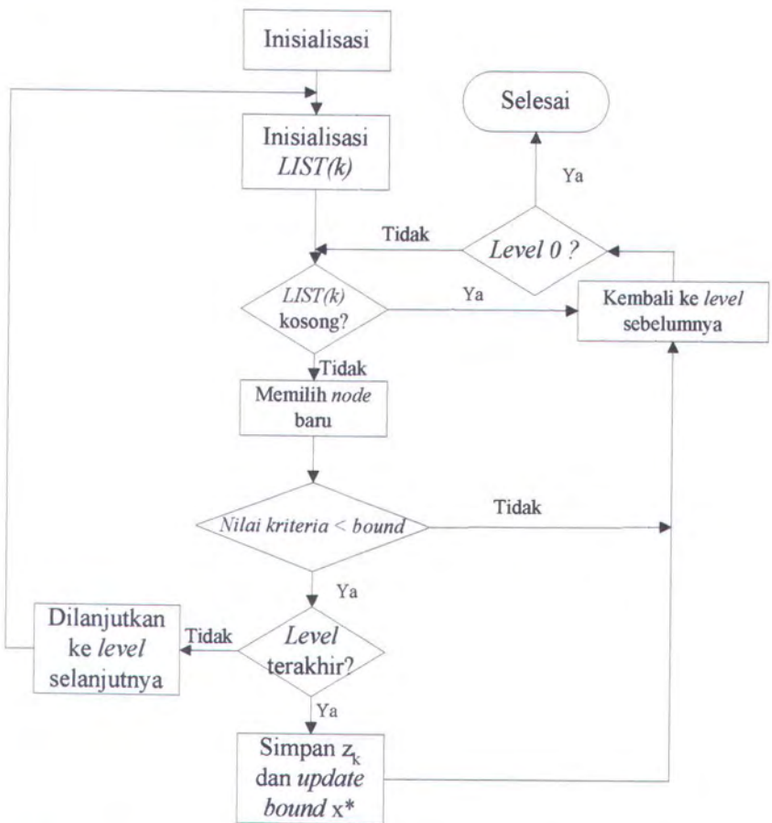
e. Langkah kelima yang disebut dengan *backtrack* ke *level* sebelumnya.

   Pada langkah ini dilakukan penelusuran balik ke *level tree* yang lebih rendah. Penelusuran balik ini bisa terjadi bila suatu *node* mengalami *cut off* atau berada pada *level* terakhir.

f. Langkah keenam yang disebut dengan *level* terakhir dan *update bound*.

   Pada langkah ini dilakukan *update* nilai *bound* dengan nilai kriteria pada *level tree* tersebut. *Update* nilai *bound* hanya terjadi pada *level* terakhir dan nilai kriteria lebih besar dari nilai *bound* sebelumnya.

Alur dari langkah-langkah algoritma *Branch and Bound* dasar tersebut secara umum digambarkan menggunakan *flowchart* yang dapat dilihat pada Gambar 4.1.



Gambar 4.1 ***Flowchart*** **Algoritma** ***Branch and Bound*** **dasar**

4.4 DESAIN ALGORITMA *IMPROVED BRANCH AND BOUND*

Secara umum pada algoritma *Improved Branch and Bound* ini terdapat 6 langkah, yaitu :

a. Inisialisasi.

Pada langkah ini variabel-variabel yang harus diinisialisasikan lebih dulu yaitu variabel *level* (k), fitur *candidate* awal $(\bar{x})$, himpunan

kontrol fitur (Ψ), jumlah dari elemen himpunan kontrol fitur (r) dan *bound*(x^*). Variabel *level* diinisialisasikan dengan nol (0). Variabel fitur *candidate* awal diinisialisasikan dengan himpunan fitur awal, sebab pada *level* nol belum ada fitur yang dihilangkan. Variabel himpunan kontrol fitur diinisialisasikan dengan himpunan fitur awal. Variabel jumlah dari elemen himpunan kontrol fitur diinisialisasikan dengan jumlah elemen himpunan fitur awal dan variabel *bound* diinisialisasikan dengan nilai paling kecil yang mungkin.

b. Langkah pertama yang disebut dengan pemilihan *node* turunan dari *current node*.

Pada langkah ini didapatkan fitur-fitur yang bisa dijadikan sebagai fitur *candidate* sesuai dengan persamaan:

$q_k = r - (D - d - k - 1)$ ....(2.12)

kemudian dibuat himpunan Q_k dan vector J_k sebagai berikut:

Semua fitur $\psi_j \in \Psi, j = 1,...,r$ diurutkan dengan syarat:

$J(\overline{X_k} \setminus \{\psi_{j1}\}) \leq J(\overline{X_k} \setminus \{\psi_{j2}\}) \leq ... \leq J(\overline{X_k} \setminus \{\psi_{jr}\})$ ....(2.13)

kemudian dipilih fitur sebanyak q_k dari himpunan fitur pada Ψ, yaitu dengan cara:

$Q_{k,i} = \psi_{j_i}$ untuk i = 1,..., q_k ....(2.14)

$J_{k,i} = J(\overline{X_k} \setminus \{\psi_{j_i}\})$ untuk i = 1,..., q_k ....(2.15)

Untuk menghindari evaluasi ganda, maka fitur ψ_{j_i} dihilangkan dari Ψ, yaitu :

$$\Psi = \Psi \setminus Q_k \qquad (2.16)$$

dan

$$r = r - q_k \qquad (2.17).$$

c. Langkah kedua yang disebut dengan pemeriksaan *node* turunan paling kanan.

Pada langkah ini nilai kriteria pada *level* ini akan diperiksa apakah lebih kecil dari nilai *bound* atau tidak. Jika nilai nilai kriteria lebih kecil dari nilai *bound* saat itu, maka akan dilanjutkan pada langkah ketiga yang menunjukkan bahwa node tersebut mengalami *cut off*. Jika nilai kriteria lebih besar dari nilai *bound* saat itu dan tidak pada *level* terakhir, maka penelusuran akan dilanjutkan ke *level* selanjutnya sesuai dengan persamaan (2.19) dan fitur *candidate* diset sesuai dengan persamaan (2.18) yaitu:

$$\overline{X_{k+1}} = \overline{X_k} \setminus \{Q_{k,q_k}\} \qquad (2.18)$$

$$k = k + 1 \qquad (2.19)$$

d. Langkah ketiga yang desebut dengan *cut off*.

Pada langkah ini terjadi *cut off* bila nilai kriteria lebih kecil dari *bound*. Fitur yang telah dihilangkan akan dikembalikan lagi pada himpunan kontrol fitur sesuai dengan persamaan:

$$\Psi = \Psi \cup \{Q_{k,q_k}\} \qquad (2.20)$$

dan

$$r = r + 1 \qquad (2.21),$$

$$Q_k = Q_k \setminus \{Q_{k,q_k}\} \qquad (2.22)$$

dan

$q_k = q_k - 1$ .....(2.23)

e. Langkah keempat yang disebut dengan *backtrack* ke *level* sebelumnya.

Pada langkah ini dilakukan penelusuran balik ke *level tree* yang lebih rendah sesuai dengan persamaan (2.24), yaitu:

$k = k - 1$ .....(2.24)

dan fitur yang telah dihilangkan akan dikembalikan pada himpunan fitur *candidate* sesuai dengan persamaan (2.25), yaitu:

$\overline{X_k} = \overline{X_{k+1}} \cup \{Q_{k,q_k}\}$ .....(2.25).

Jika *level tree* bernilai -1, maka algoritma berhenti melakukan penelusuran.

f. Langkah kelima yang disebut dengan *update* nilai *bound*.

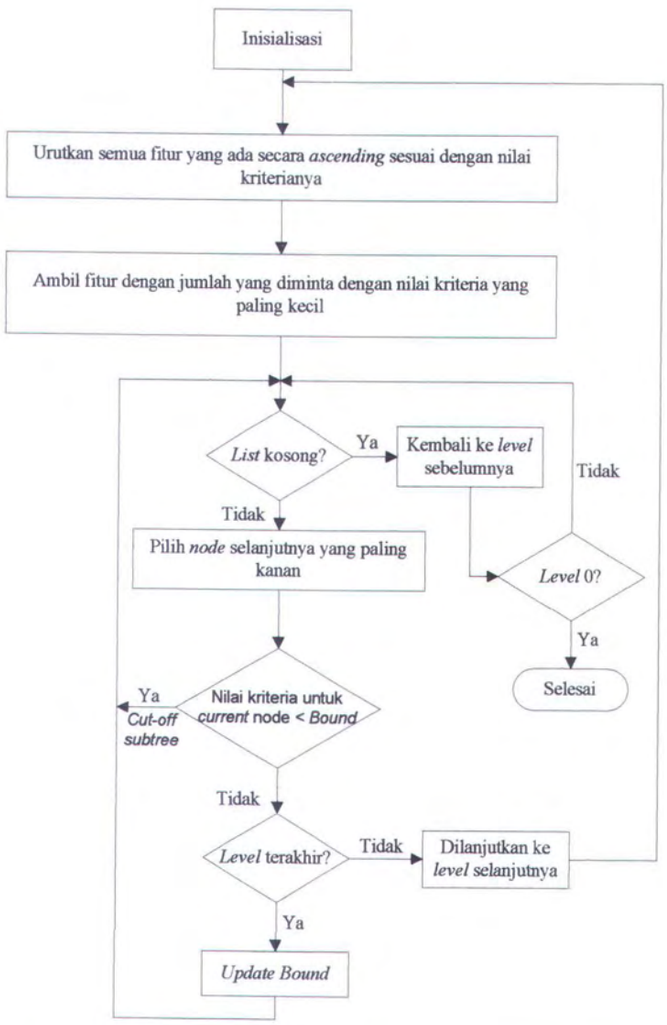
Pada langkah ini nilai *bound* di-*update* dengan nilai kriteria pada *level tree* tersebut sesuai dengan persamaan (2.26), yaitu:

$x^* = J_{k,q_k}$ .....(2.26)

dan fitur terbaik saat itu disimpan sesuai dengan persamaan (2.27), yaitu:

$X = \overline{X_{k+1}}$ .....(2.27).

Alur dari langkah-langkah algoritma *Improved Branch and Bound* tersebut secara umum digambarkan menggunakan *flowchart* yang dapat dilihat pada Gambar 4.2.



Gambar 4.2 *Flowchart* Algoritma *Improved Branch and Bound*

4.5 DESAIN ALGORITMA *BRANCH AND BOUND PARTIAL PREDICTION*

Secara umum pada algoritma *Branch and Bound Partial Prediction* ini terdapat 6 langkah, yaitu :

a. Inisialisasi.

Pada langkah ini variabel-variabel yang harus diinisialisasikan lebih dulu yaitu variabel *level* (k), fitur *candidate* awal $(\bar{x})$, himpunan kontrol fitur (Ψ), jumlah dari elemen himpunan kontrol fitur (r) dan *bound*(x^*). Variabel *level* diinisialisasikan dengan nol (0). Variabel fitur *candidate* awal diinisialisasikan dengan himpunan fitur awal, sebab pada *level* nol belum ada fitur yang dihilangkan. Variabel himpunan kontrol fitur diinisialisasikan dengan himpunan fitur awal. Variabel jumlah dari elemen himpunan kontrol fitur diinisialisasikan dengan jumlah elemen himpunan fitur awal dan variabel *bound* diinisialisasikan dengan nilai paling kecil yang mungkin.

b. Langkah pertama yang disebut dengan pemilihan *node* turunan dari *current node*.

Pada langkah ini didapatkan fitur-fitur yang bisa dijadikan sebagai fitur *candidate*. Pada langkah ini dilakukan suatu prediksi sederhana untuk menentukan fitur-fitur yang dijadikan sebagai fitur *candidate*. Jumlah dari banyaknya fitur *candidate* didapatkan dengan persamaan (3.5), yaitu:

$$q_k = r - (D - d - k - 1) \quad \text{...(3.5).}$$

Fitur-fitur diurutkan secara *descending* didasarkan pada nilai prediksi tiap fitur yang telah diperoleh sesuai dengan persamaan (3.6), yaitu:

$$A_{\psi_{j_1}} \geq A_{\psi_{j_2}} \geq \ldots \geq A_{\psi_{j_r}} \quad \text{...(3.6).}$$

Hal ini berbeda dengan algoritma *Improved Branch and Bound* yang

mengurutkan fitur secara *ascending* berdasarkan pada nilai kriteria sebenarnya. Setelah fitur-fitur diurutkan, selanjutnya dipilih jumlah fitur yang diminta dan nilai kriteria sebenarnya dari fitur yang dipilih dihitung sesuai dengan persamaan (3.7) dan (3.8), yaitu:

$$Q_{k,i} = \psi_{j_i} \quad \text{.......................................} (3.7)$$

untuk i = 1,...,q_k

$$J_{k,i} = J\left(\overline{X_k} \setminus \{\psi_{j_i}\}\right) \quad \text{.......................................} (3.8)$$

untuk i = 1,...,q_k.

Fitur-fitur yang telah dijadikan fitur *candidate* dihilangkan dari himpunan kontrol fitur sesuai dengan persamaan (3.9) dan (3.10), yaitu:

$$\Psi = \Psi \setminus Q_k \quad \text{.......................................} (3.9)$$

dan

$$r = r - q_k \quad \text{.......................................} (3.10).$$

c. Langkah kedua yang disebut dengan pemeriksaan *node* turunan paling kanan.

Pada langkah ini nilai kriteria pada *level* ini akan diperiksa apakah lebih kecil dari nilai *bound* atau tidak. Jika nilai kriteria lebih kecil dari nilai *bound* saat itu, maka akan dilanjutkan pada langkah ketiga yang menunjukkan bahwa node tersebut mengalami *cut off*. Jika nilai kriteria lebih besar dari nilai *bound* saat itu dan tidak pada *level* terakhir, maka penelusuran akan dilanjutkan ke *level* selanjutnya sesuai dengan persamaan (3.12), yaitu:

$k = k + 1$ .....(3.12)

dan fitur *candidate* diset sesuai dengan persamaan (3.11), yaitu:

$\overline{X_{k+1}} = \overline{X_k} \setminus \{Q_{k,q_k}\}$ .....(3.11).

d. Langkah ketiga yang disebut dengan *cut off.*

Pada langkah ini terjadi *cut off* bila nilai kriteria lebih kecil dari *bound.* Fitur yang telah dihilangkan akan dikembalikan lagi pada himpunan kontrol fitur sesuai dengan persamaan :

$\Psi = \Psi \cup \{Q_{k,q_k}\}$ .....(3.13)

dan

$r = r + 1$ .....(3.14),

$Q_k = Q_k \setminus \{Q_{k,q_k}\}$ .....(3.15)

dan

$q_k = q_k - 1$ .....(3.16).

e. Langkah keempat yang disebut dengan *backtrack* ke *level* sebelumnya.

Pada langkah ini dilakukan penelusuran balik ke *level tree* yang lebih rendah sesuai dengan persamaan (3.17), yaitu:

$k = k - 1$ .....(3.17)

dan fitur yang telah dihilangkan akan dikembalikan pada himpunan fitur *candidate* sesuai dengan persamaan (3.18), yaitu:

$$\overline{X_k} = \overline{X_{k+1}} \cup \{Q_{k,q_k}\} \quad \text{.....(3.18).}$$

Jika *level tree* bernilai -1, maka algoritma berhenti melakukan penelusuran.

f. Langkah kelima yang disebut dengan *update* nilai *bound.*

Pada langkah ini nilai *bound* di-*update* dengan nilai kriteria pada *level tree* tersebut sesuai dengan persamaan (3.19), yaitu:

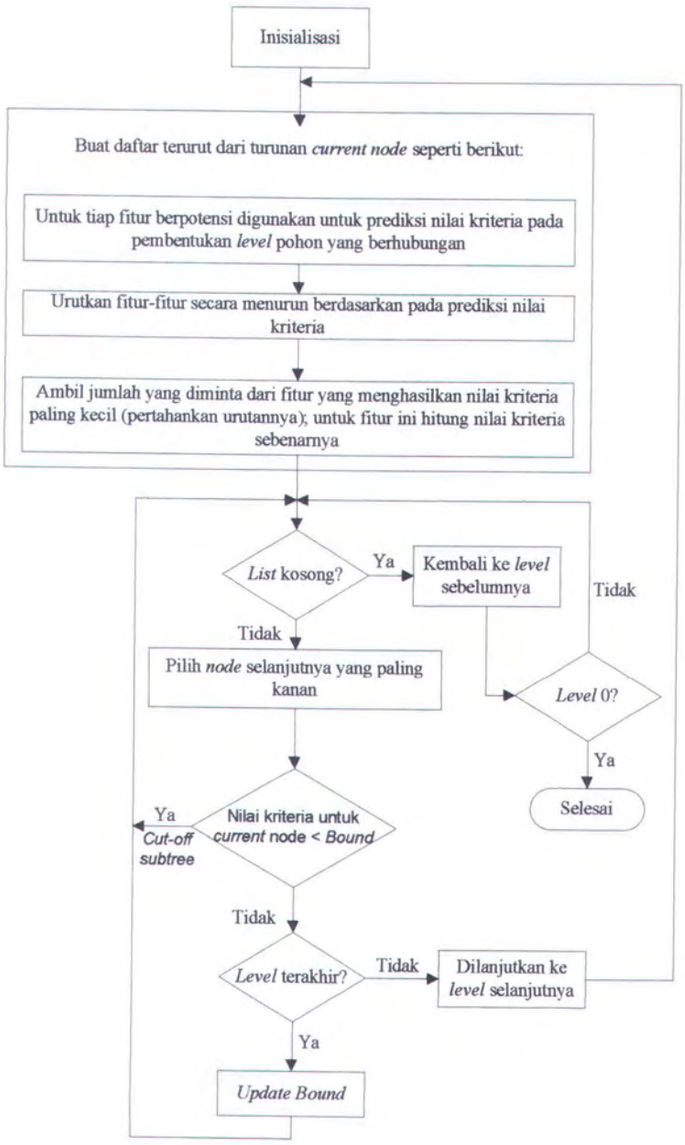
$$x^* = J_{k,q_k} \quad \text{.....(3.19)}$$

dan fitur terbaik saat itu disimpan sesuai dengan persamaan (3.20), yaitu:

$$X = \overline{X_{k+1}} \quad \text{.....(3.20).}$$

Keenam langkah tersebut secara umum sama dengan langkah-langkah pada algoritma *Improved Branch and Bound.* Langkah pertama saja pada algoritma ini yang berbeda dengan langkah pertama pada algoritma *Improved Branch and Bound.*

Langkah kedua sampai langkah kelima pada algoritma BBPP ini sama dengan langkah kedua sampai langkah kelima algoritma *Improved Branch and Bound.* Alur dari langkah-langkah algoritma *Branch and Bound Partia Prediction* tersebut secara umum digambarkan menggunakan *flowchart* yang dapat dilihat pada Gambar 4.3.



Gambar 4.3 ***Flowchart*** **Algoritma** ***Branch and Bound Partial Prediction***

4.6 DESAIN ALGORITMA *FAST BRANCH AND BOUND*

Secara umum pada algoritma *Fast Branch and Bound* ini terdapat 6 langkah, yaitu :

a. Inisialisasi.

Pada langkah ini variabel-variabel yang harus diinisialisasikan lebih dulu yaitu variabel *level* (k), fitur *candidate* awal $(\bar{x})$, himpunan kontrol fitur (Ψ), jumlah dari elemen himpunan kontrol fitur (r) dan $bound(x^*)$. Variabel *level* diinisialisasikan dengan nol (0). Variabel fitur *candidate* awal diinisialisasikan dengan himpunan fitur awal, sebab pada *level* nol belum ada fitur yang dihilangkan. Variabel himpunan kontrol fitur diinisialisasikan dengan himpunan fitur awal. Variabel jumlah dari elemen himpunan kontrol fitur diinisialisasikan dengan jumlah elemen himpunan fitur awal dan variabel *bound* diinisialisasikan dengan nol (0).

b. Langkah pertama yang disebut dengan pemilihan *node* turunan dari *current node*.

Pada langkah ini didapatkan fitur-fitur yang bisa dijadikan sebagai fitur *candidate*. Jumlah dari banyaknya fitur *candidate* didapatkan dengan persamaan (3.23), yaitu:

$q_k = r - (D - d - k - 1)$ ....(3.23).

Fitur-fitur diurutkan secara *ascending* sesuai dengan persamaan (3.26) dimana nilainya didapatkan sesuai dengan persamaan (3.24) dan (3.25), yaitu:

$v_j = J_{k-1,q_{k-1}} - A_{\psi_j}$ ....(3.24)

$v_j = J(\overline{x_k} \setminus \{\psi_j\})$ ....(3.25)

$v_{j_1} \le v_{j_2} \le ... \le v_{j_r}$ .....(3.26).

Setelah fitur-fitur diurutkan, selanjutnya dipilih jumlah fitur yang diminta dan nilai kriteria sebenarnya dari fitur yang dipilih dihitung sesuai dengan persamaan (3.27), (3.28), (3.29), (3.30) dan (3.31), yaitu:

$Q_{k,i} = \psi_{j_i}$ .....(3.27)

dan

$J_{k,i} = v_{j_i}$ .....(3.28),

jika v_{j_i} merupakan sebuah rekord yang nilainya dihitung dan

$J_{k,i} = J_{k-1,q_{k-1}} - \gamma . A_{\psi_{j_i}}$ .....(3.29),

jika v_{j_i} merupakan sebuah rekord yang nilainya diprediksi.

$T_{k,i} = "C"$ .....(3.30),

jika v_{j_i} merupakan sebuah rekord yang nilainya dihitung dan

$T_{k,i} = "P"$ .....(3.31)

jika v_{j_i} merupakan sebuah rekord yang nilainya diprediksi.

Fitur-fitur yang telah dijadikan fitur *candidate* dihilangkan dari himpunan kontrol fitur sesuai dengan persamaan (3.32) dan (3.33), yaitu:

$\Psi = \Psi \setminus Q_k$ .....(3.32)

dan

$r = r - q_k$ .....(3.33).

c. Langkah kedua yang disebut dengan pemeriksaan *node* turunan paling kanan.

Pada langkah ini akan diperiksa apakah nilai kriteria lebih kecil dari *bound* dan vektor tipenya "P", maka nilai kriteria sebenarnya harus dihitung sesuai dengan persamaan (3.34) dan (3.35), yaitu:

$$J_{k,q_k} = J(\overline{x_k} \setminus \{Q_{k,q_k}\}) \quad \text{.....(3.34)}$$

dan

$$T_{k,q_k} = "C" \quad \text{.....(3.35).}$$

Jika nilai kriterianya lebih kecil dari *bound* dan vektor tipenya "C", maka akan dilanjutkan ke langkah ketiga untuk dilakukan *cut off* pada *node* tersebut. Jika nilai kriteria lebih besar dari nilai *bound* saat itu dan tidak pada *level* terakhir, maka penelusuran akan dilanjutkan ke *level* selanjutnya sesuai dengan persamaan (3.37), yaitu:

$$k = k + 1 \quad \text{.....(3.37)}$$

dan fitur *candidate* diset sesuai dengan persamaan (3.36), yaitu:

$$\overline{X_{k+1}} = \overline{X_k} \setminus \{Q_{k,q_k}\} \quad \text{.....(3.36).}$$

d. Langkah ketiga yang desebut dengan *cut off.*

Pada langkah ini terjadi *cut off* bila nilai kriteria lebih kecil dari *bound.* Fitur yang telah dihilangkan akan dikembalikan lagi pada himpunan kontrol fitur sesuai dengan persamaan (3.38), (3.39), (3.40) dan (3.41), yaitu:

$$\Psi = \Psi \cup \{Q_{k,q_k}\} \quad \text{.....(3.38)}$$

dan

$r = r + 1$ .....(3.39),

$Q_k = Q_k \setminus \{Q_{k,q_k}\}$ .....(3.40)

dan

$q_k = q_k - 1$ .....(3.41).

e. Langkah keempat yang disebut dengan *backtrack* ke *level* sebelumnya.

Pada langkah ini dilakukan penelusuran balik ke *level tree* yang lebih rendah sesuai dengan persamaan (3.42), yaitu:

$k = k - 1$ .....(3.42)

dan fitur yang telah dihilangkan akan dikembalikan pada himpunan fitur *candidate* sesuai dengan persamaan (3.43), yaitu:

$\overline{X_k} = \overline{X_{k+1}} \cup \{Q_{k,q_k}\}$ .....(3.43).

Jika *level tree* bernilai -1, maka algoritma berhenti melakukan penelusuran.

f. Langkah kelima yang disebut dengan *update* nilai *bound.*

Pada langkah ini nilai *bound* di-*update* dengan nilai kriteria pada *level tree* tersebut sesuai dengan persamaan (3.44), yaitu:

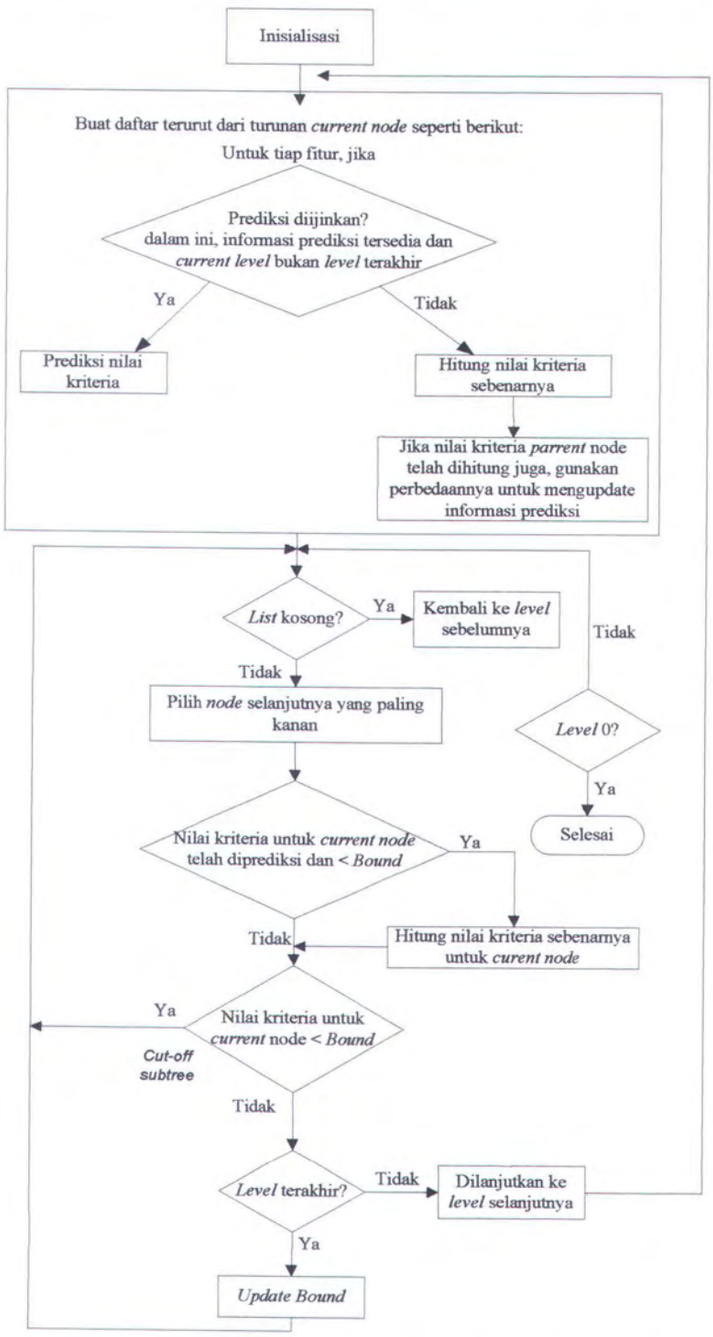
$x^* = J_{k,q_k}$ .....(3.44)

dan fitur terbaik saat itu disimpan sesuai dengan persamaan (3.45), yaitu:

$X = \overline{X_{k+1}}$ .....(3.45).

Langkah kedua sampai langkah kelima pada algoritma FBB ini sama dengan langkah kedua sampai langkah kelima algoritma *Improved Branch and Bound.*

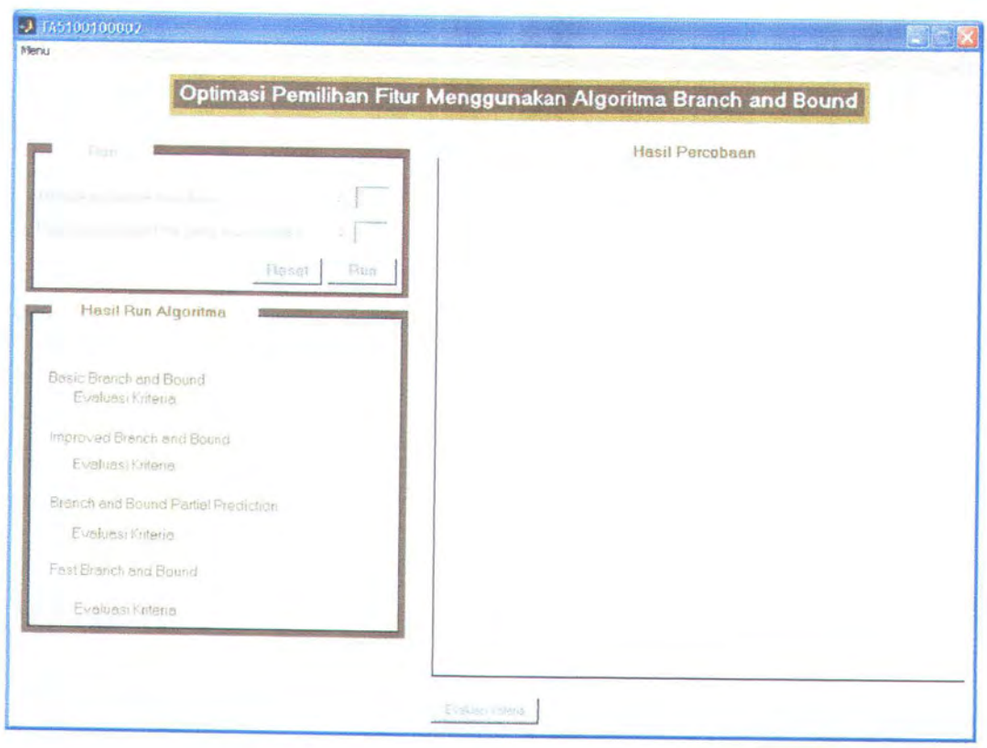
Alur dari langkah-langkah algoritma *Fast Branch and Bound* tersebut secara umum digambarkan menggunakan *flowchart* yang dapat dilihat pada Gambar 4.4.



Gambar 4.4 *Flowchart* Algoritma *Fast Branch and Bound*

4.7 ANTARMUKA

Form antarmuka pada tugas akhir ini ada satu bagian. Tampilan awal aplikasi dapat dilihat pada Gambar 4.5. Pada tampilan awal ini terdapat menu pilihan yang bisa dipilih. Pada menu pilihan ini berisi tiga menu yaitu menu run algoritma, menu hasil percobaan dan menu untuk keluar dari aplikasi. Tampilan menu ini dapat dilihat pada Gambar 4.6 dan 4.7.

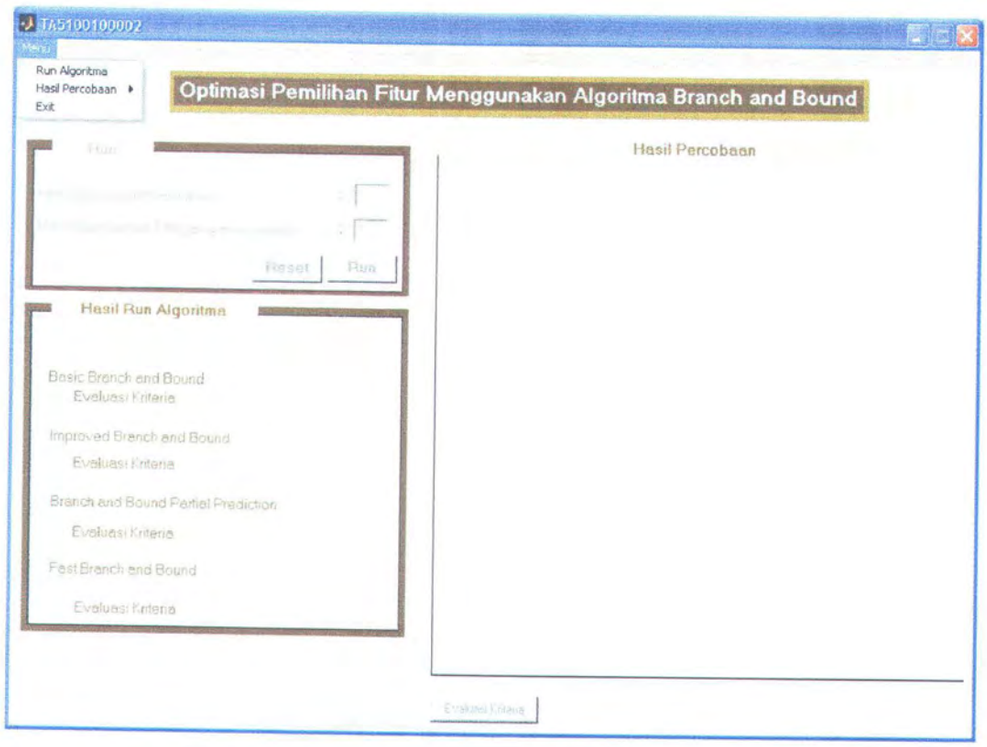


Gambar 4.5 Form awal aplikasi optimasi pemilihan fitur

Penjelasan kegunaan fitur-fitur yang ada dalam aplikasi adalah sebagai berikut:

1. Menu Editor "Menu"

   digunakan untuk memilih menu-menu pilihan yang ada dalam aplikasi. Menu-menu pilihan yang disediakan adalah sebagai berikut:



Gambar 4.6 Menu-menu pada form aplikasi optimasi pemilihan fitur

a. "Run Algoritma"

digunakan untuk menjalankan keempat algoritma *Branch and Bound.* Tampilan untuk run algoritma ini dapat dilihat pada Gambar 4.8.

b. "Hasil Percobaan"

digunakan untuk memilih percobaan mana saja yang akan ditampilkan. Tampilan untuk hasil percobaan ini dapat dilihat pada Gambar 4.9.

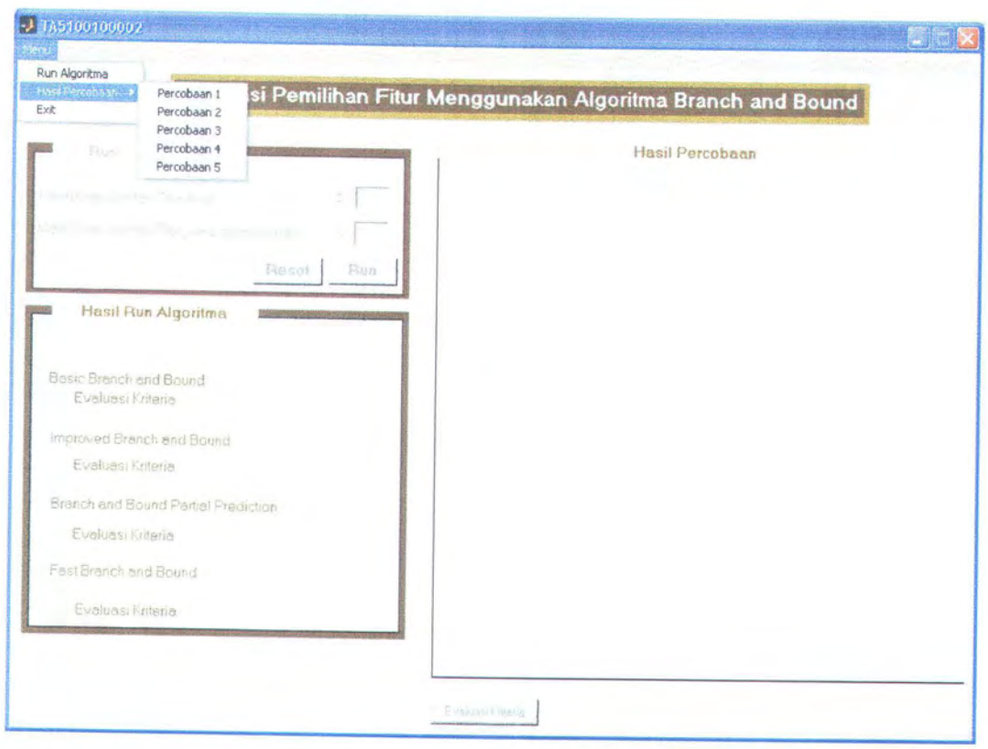
Menu-menu pada menu "Hasil Percobaan" adalah sebagai berikut:

- "Percobaan 1"

digunakan untuk menampilkan hasil grafik dari percobaan 1 yang telah dilakukan.

- "Percobaan 2"

  digunakan untuk menampilkan hasil grafik dari percobaan 2 yang telah dilakukan.



Gambar 4.7 Menu-menu hasil percobaan pada form aplikasi optimasi pemilihan fitur

- "Percobaan 3"

  digunakan untuk menampilkan hasil grafik dari percobaan 3 yang telah dilakukan.

- "Percobaan 4"

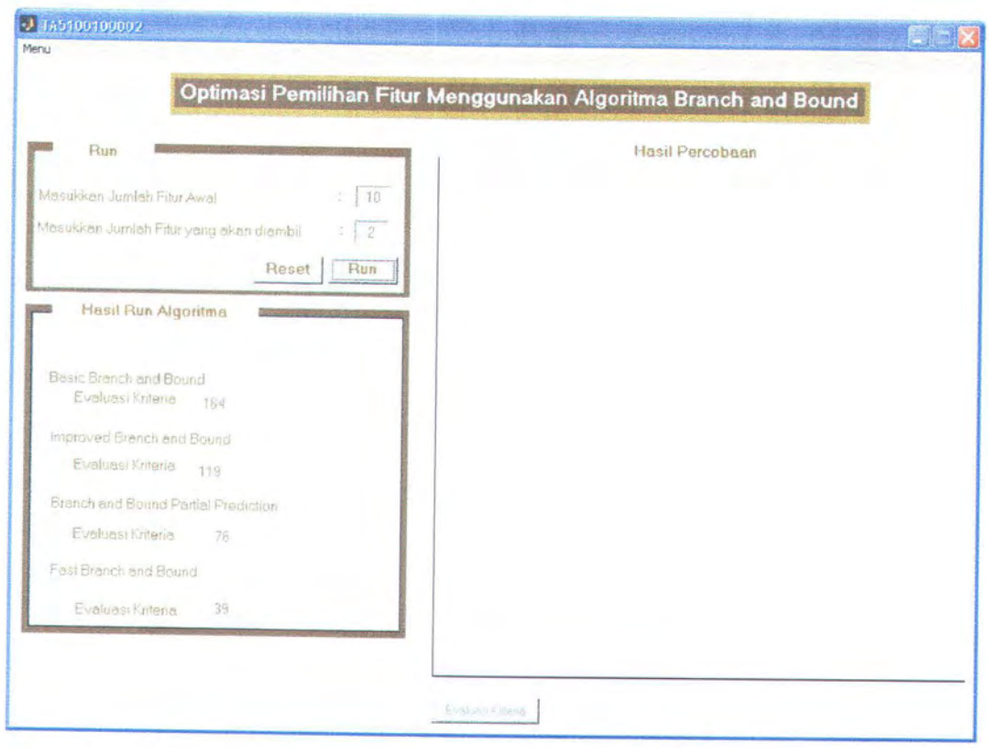
digunakan untuk menampilkan hasil grafik dari percobaan 4 yang telah dilakukan.

- "Percobaan 5"

  digunakan untuk menampilkan hasil grafik dari percobaan 5 yang telah dilakukan.

c. "Exit"

digunakan untuk keluar dari aplikasi.



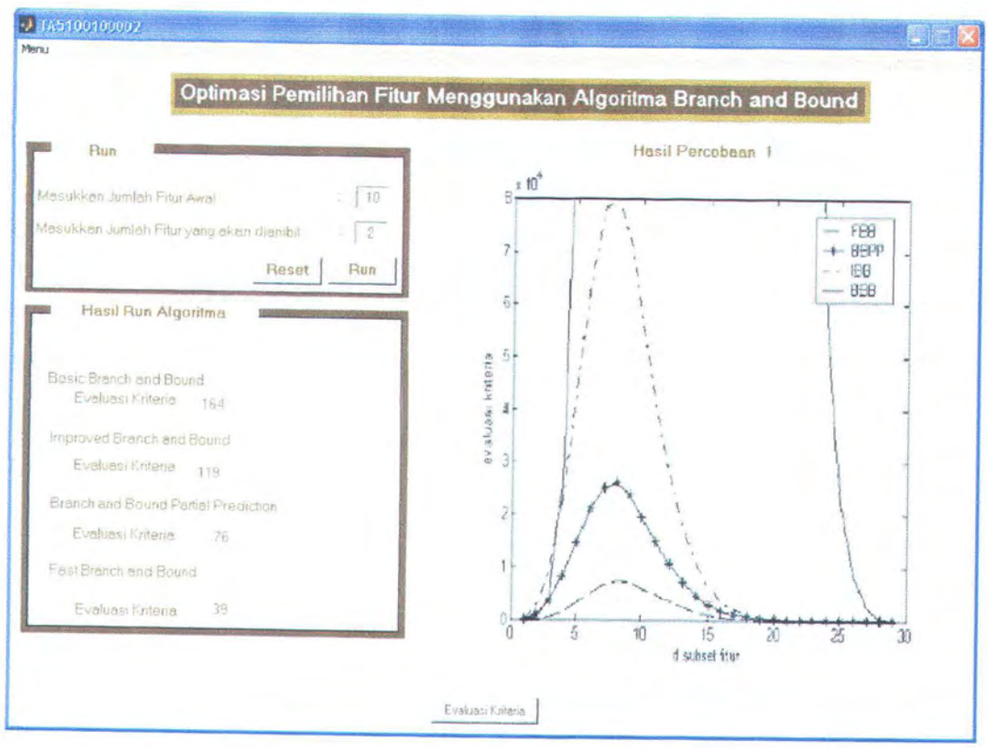
Gambar 4.8 Tampilan menu run algoritma pada form aplikasi optimasi pemilihan fitur

2. Kotak teks "Masukkan jumlah fitur awal"

   digunakan untuk meminta masukan jumlah fitur awal.

3. Kotak teks "Masukkan jumlah fitur yang diambil"

digunakan untuk meminta masukan jumlah fitur yang akan diambil.



Gambar 4.9 Tampilan hasil percobaan pada form aplikasi optimasi pemilihan fitur

4. Tombol "Reset"

   digunakan untuk menghapus semua masukan dan keluaran dari "Run Algoritma"

5. Tombol "Run"

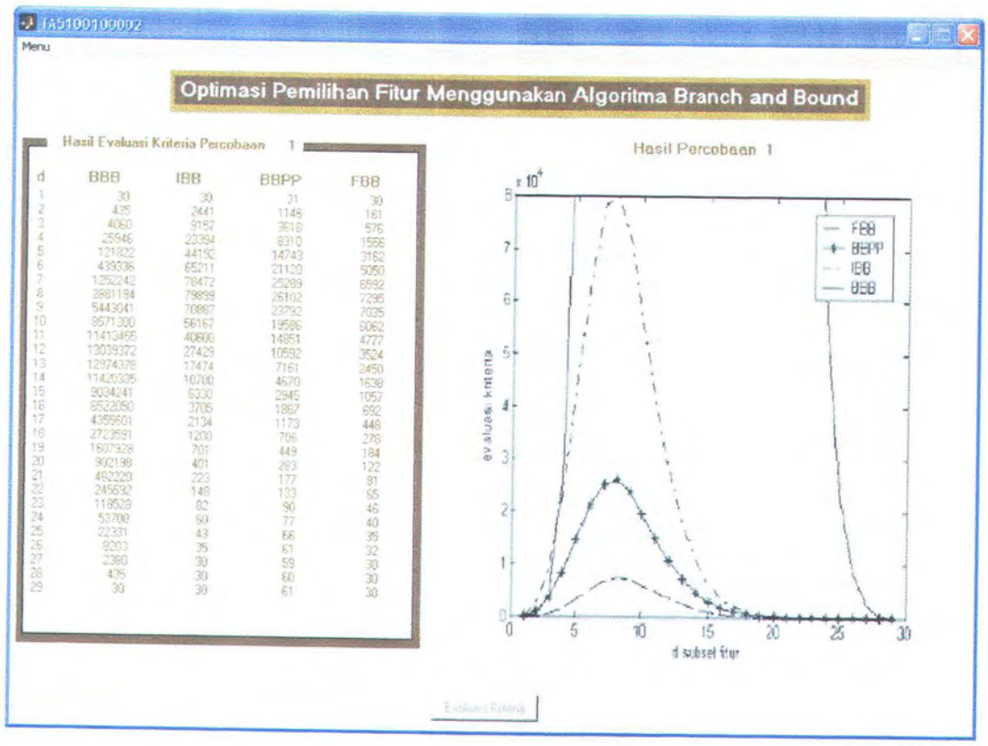
   digunakan untuk menjalankan algoritma sesuai dengan masukan yang telah dimasukkan.

6. Frame "Hasil Run Algoritma"

   digunakan untuk menampilkan hasil evaluasi kriteria yang terjadi pada keempat algoritma *Branch and Bound.*

7. Tombol "Evaluasi Kriteria"

digunakan untuk menampilkan hasil evaluasi fungsi kriteria pada percobaan yang telah dipilih. Tampilan untuk menampilkan evaluasi kriteria pada percobaan yang telah dilakukan dapat dilihat pada Gambar 4.10.



Gambar 4.10 Tampilan evaluasi kriteria pada form aplikasi optimasi pemilihan fitur

BAB V

IMPLEMENTASI PERANGKAT LUNAK

Pada bab ini dijelaskan mengenai implementasi algoritma *Branch and Bound* dasar, algoritma *Improved Branch and Bound*, algoritma *Branch and Bound Partial Prediction* dan algoritma *Fast Branch and Bound* pada pemilihan fitur. Pembahasan ini meliputi lingkungan implementasi perangkat lunak, serta *pseudocode* program untuk setiap proses.

5.1 LINGKUNGAN IMPLEMENTASI

Pada bagian ini akan dijelaskan mengenai lingkungan pembangunan aplikasi yang meliputi perangkat lunak dan perangkat keras yang digunakan. Spesifikasi perangkat lunak dan perangkat keras yang digunakan dalam pembangunan aplikasi optimasi pemilihan fitur ini dapat dilihat pada tabel 5.1.

Tabel 5.1 Lingkungan Pembangunan Aplikasi

Perangkat Keras	Prosesor : Intel Pentium IV 2.80 GHz Memori : 1 GB
Perangkat Lunak	Sistem Operasi : Windows Perangkat Lunak Pembangun : Matlab 6.5.1

5.2 ALGORITMA *BRANCH AND BOUND* DASAR

Proses yang terjadi pada algoritma *Branch and Bound* dasar pada pemilihan fitur adalah sebagai berikut :

1. Inisialisasi
2. *Generate successor* (inisialisasi *LIST(k)*)
3. Pemilihan *node* baru
4. *Check bound*
5. *Backtrack* ke *level* sebelumnya
6. *Level* terakhir (*update bound*)

Selanjutnya akan diberikan *pseudocode* untuk setiap proses diatas.

5.2.1 Inisialisasi

Pada langkah ini variabel-variabel yang harus diinisialisasikan lebih dulu yaitu variabel $bound(x^*)$, *level* dan z_0. Variabel *bound* diinisialisasikan dengan nol (0), variabel *level* (k) diinisialisasikan dengan 1 dan variabel z_0 diinisialisasikan dengan nol (0). *Pseudocode* untuk proses inisialisasi dapat dilihat pada Gambar 5.1.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                  step 1                      %
%               inisialization                 %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
D=5; % number of set fitur
d=2; % number of subset fitur
dbar=D-d;     % number of remove fitur
k=1;          % level tree
bound=0;      % bound
z(dbar+1,:)=0;  % z0 inisialization
subset_fitur=[1:D];
criterion_evaluation=0;
LIST=[];
done=1;
step=2;
```

Gambar 5.1 Kode program inisialisasi algoritma BBB

5.2.2 *Generate successor* (inisialisasi *LIST(k)*)

Pada langkah ini didapatkan fitur-fitur yang bisa dijadikan sebagai fitur *candidate*. Fitur *candidate* ini disimpan pada suatu variabel $LIST(k)$. Fitur *candidate* ini merupakan fitur-fitur yang berpotensi untuk dihilangkan. *Pseudocode* untuk proses *Generate successor* (inisialisasi *LIST(k)*) dapat dilihat pada Gambar 5.2.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                     step 2                   %
%               generate successor             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if k==1
    LIST(k,:)=[(z(dbar+1,:)+1):(d+k)];
else
    candidate_fitur_temporary=[(z(k-1,:)+1):(d+k)];
    [row_candidate_fitur_temporary,column_candidate_fitur_temporary]=
         size(candidate_fitur_temporary);
    LIST(k,[1:column_candidate_fitur_temporary])=candidate_fitur_temporary;
end
```

Gambar 5.2 Kode program *generate successor* (inisialisasi *LIST(k)*) algoritma BBB

5.2.3 Pemilihan *node* baru

Pada langkah ini satu fitur akan dipilih dari fitur *candidate* untuk dijadikan sebagai *node* pada *level tree* yang berhubungan. Setelah satu fitur pada fitur *candidate* dijadikan *node* pada *tree*, maka fitur tersebut akan dihilangkan dari fitur *candidate*. *Pseudocode* program pemilihan *node* baru dapat dilihat pada Gambar 5.3.

5.2.4 *Check bound*

Pada langkah ini setiap nilai kriteria pada *level tree* dihitung dan diperiksa untuk dibandingkan dengan nilai *bound* saat itu, apakah lebih kecil atau tidak. Bila nilai kriteria lebih kecil dari nilai *bound* saat itu, maka akan dilanjutkan pada langkah kelima yang menunjukkan bahwa *node* tersebut mengalami *cut off*. *Pseudocode*

program *check bound* pada *Branch and Bound* dasar ini dapat dilihat pada Gambar 5.4.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                    step 3                    %
%                select new node               %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
list=LIST(k,:);
check_list=(nonzeros(list))';
flag_check_list=isempty(check_list);
if flag_check_list==1 % list empty
    step=5;
else
    subset_fitur=[subset_fitur,z(k)];
    z(k,:)=max(list);
    list=(nonzeros(list))';
    list=setdiff(list,z(k,:));
    [row_list,column_list]=size(list);
    LIST(k,[1:column_list])=list;
    LIST(k,column_list+1)=0;
    step=4;
end
```

Gambar 5.3 Kode program pemilihan *node* baru algoritma BBB

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                    step 4                    %
%                  check bound                 %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if k==1
    subset_fitur=[1:D];
end
    subset_fitur=(nonzeros(setdiff(subset_fitur,z(k,:))))';
    [creterion_value]=nilai_kriteria4(subset_fitur);
    criterion_evaluation=criterion_evaluation+1;
    if (creterion_value<=bound)
        step=5;
    elseif k==dbar
        step=6;
    else
        k=k+1 ;
        step=2;
    end
```

Gambar 5.4 Kode program *check bound* algoritma BBB

5.2.5 *Backtrack* ke *level* sebelumnya

Pada langkah ini dilakukan penelusuran balik ke *level tree* yang lebih rendah. Penelusuran balik ini bisa terjadi bila suatu *node* mengalami *cut off* atau berada pada *level* terakhir. Bila *level* didapatkan nilai -1, maka algoritma akan berhenti melakukan penelusuran. *Pseudocode* program *backtrack* ke *level* sebelumnya pada *Branch and Bound* dasar ini dapat dilihat pada Gambar 5.5.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                    step 5                     %
%                 backtracking                  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
list=LIST(k,:);
check_list=nonzeros(list);
check_list=check_list';
flag_check_list=isempty(check_list);
if flag_check_list==1 %list empty
    k=k-1;
    subset_fitur=(nonzeros([subset_fitur,z(k+1)]))';
    z(k+1)=0;
    if k==0
        done=0;
        break
    else
        step=3;
    end
else
    step=3;
end
```

Gambar 5.5 Kode program *backtrack* ke level sebelumnya algoritma BBB

5.2.6 *Level* terakhir (*update bound*)

Pada langkah ini dilakukan *update* nilai *bound* dengan nilai kriteria pada *level tree* tersebut. *Update* nilai *bound* hanya terjadi pada *level* terakhir dan nilai kriteria lebih besar dari nilai *bound* sebelumnya. *Pseudocode* program *level* terakhir (*update bound*) pada *Branch and Bound* dasar ini dapat dilihat pada Gambar 5.6.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                  step 6                          %
%          last level and update bound             %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
bound=creterion_value;
x=subset_fitur;
remove_fitur=[z(1:dbar)];
step=5;
```

Gambar 5.6 Kode program *level* terakhir (*update bound*) algoritma BBB

5.3 ALGORITMA *IMPROVED BRANCH AND BOUND*

Terdapat enam proses pada algoritma *improved branch and bound*, yaitu:

1. Inisialisasi
2. Pemilihan *node* turunan dari *current node*
3. Pemeriksaan *node* turunan paling kanan
4. *Cut-off*
5. *Backtrack* ke *level* sebelumnya
6. *Update* nilai *bound*
7. *Update* vektor kontribusi dan vektor kounter

Selanjutnya akan diberikan *pseudocode* untuk setiap proses diatas.

5.3.1 Inisialisasi

Pada langkah ini variabel-variabel yang harus diinisialisasikan lebih dulu yaitu variabel *level* (k), fitur candidate awal $(\bar{x})$, himpunan kontrol fitur (Ψ), jumlah dari elemen himpunan kontrol fitur (r) dan *bound* (x^*). Variabel *level* diinisialisasikan dengan nol (0). Variabel fitur *candidate* awal diinisialisasikan dengan himpunan fitur awal, sebab pada level nol ini belum ada fitur yang dihilangkan. Variabel himpunan kontrol fitur diinisialisasikan dengan himpunan

fitur awal. Variabel jumlah dari elemen himpunan kontrol fitur diinisialisasikan dengan jumlah elemen himpunan fitur awal dan variabel bound diinisialisasikan dengan nilai paling kecil yang mungkin. *Pseudocode* untuk proses inisialisasi dapat dilihat pada Gambar 5.7.

```
1   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2   %                  Inisialization             %
3   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
4   D=5; % number of set fitur
5   d=2; % number of subset fitur
6   dbar=D-d; % number of remove fitur
7   k=0;      % level tree
8   r=D;
9   Y=[1:D];
10  xbar(dbar+1,:)=Y;
11  PSI=Y;
12  done=1;
13  step=1;
14  criterion_evaluation_temporary=0;
15  criterion_evaluation=0
16  row_criterion_evaluation_matrix=1;
17  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
18  %first bound initialization
19  boundawal=PSI(1:d);
20  tempbound=nilai_kriteria6(boundawal);
21  bound=tempbound;
22  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Gambar 5.7 Kode program inisialisasi algoritma IBB

5.3.2 Pemilihan *Node* Turunan dari *Current Node*

Pada langkah ini didapatkan fitur-fitur yang bisa dijadikan sebagai fitur *candidate* sesuai dengan persamaan (2.12), yaitu:

$$q_k = r - (D - d - k - 1) \quad \text{...} (2.12)$$

dengan *pseudocode* program dapat dilihat pada baris ke 7 dan 22, persamaan (2.13), yaitu:

$J(\overline{X_k} \setminus \{\psi_{j1}\}) \le J(\overline{X_k} \setminus \{\psi_{j2}\}) \le \ldots \le J(\overline{X_k} \setminus \{\psi_{jr}\})$ ....(2.13)

dengan *pseudocode* program dapat dilihat pada baris ke 56 sampai 66, persamaan (2.14) dan (2.15), yaitu:

$Q_{k,i} = \psi_{j_i}$ untuk i = 1,..., q_k ....(2.14)

$J_{k,i} = J(\overline{X_k} \setminus \{\psi_{j_i}\})$ untuk i = 1,..., q_k ....(2.15)

dengan *pseudocode* program dapat dilihat pada baris ke 72 sampai 78, persamaan (2.16) dan (2.17), yaitu:

$\Psi = \Psi \setminus Q_k$ ....(2.16)

dan

$r = r - q_k$ ....(2.17)

dengan *pseudocode* program dapat dilihat pada baris ke 83 sampai 94. *Pseudocode* untuk proses pemilihan *node* turunan dari *current node* secara keseluruhan dapat dilihat pada Gambar 5.8, 5.9 dan 5.10.

```
1   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2   %                       step 1                         %
3   %      select descendants of the current node         %
4   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
5   %for k=0
6   if k==0
7       q(dbar+1,1)=r-(D-d-k-1);
8       for j=1:r
9           psi=PSI(1,j);
10          cekfitur=setdiff(PSI,psi);
11          J(dbar+1,j)=nilai_kriteria6(cekfitur);
12  criterion_evaluation_temporary=criterion_evaluation_temporary+1;
13          J_remove_feature(j,:)=[psi,J(dbar+1,j)];
14          psi=[];
15      end
16      criterion_evaluation_matrix(row_criterion_evaluation_matrix,1)=
17                                  [criterion_evaluation_temporary];
```

Gambar 5.8 Kode program pemilihan *node* turunan dari *current node* algoritma IBB

```
        criterion_evaluation_temporary=0;
        row_criterion_evaluation_matrix=row_criterion_evaluation_matrix+1;
    %another
    else
        q(k,1)=r-(D-d-k-1);
        for j=1:r
            psi=PSI(1,j);
            cekfitur=(nonzeros(setdiff(xbar(k,:),psi)))';
            J(k,j)=nilai_kriteria6(cekfitur);
    criterion_evaluation_temporary=criterion_evaluation_temporary+1;
            J_remove_feature(j,:)=[psi,J(k,j)];
            [rcekfitur,ccekfitur]=size(cekfitur);
            psi=[];
        end
        criterion_evaluation_matrix(row_criterion_evaluation_matrix,1)=
    [criterion_evaluation_temporary];
        criterion_evaluation_temporary=0;
        row_criterion_evaluation_matrix=row_criterion_evaluation_matrix+1;
    end
    sd=row_criterion_evaluation_matrix-1
    if sd==1
        criterion_evaluation=criterion_evaluation
    else
        criterion_evaluation1=criterion_evaluation_matrix(sd-1,1);
        criterion_evaluation2=criterion_evaluation_matrix(sd,1);
        temp=criterion_evaluation1-criterion_evaluation2;
        if temp<0
            criterion_evaluation=criterion_evaluation+criterion_evaluation1
            criterion_evaluation=criterion_evaluation+1
        else
            criterion_evaluation=criterion_evaluation+temp
        end
    end
    J_remove_feature=J_remove_feature';
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % sort the current true criterion value decrease
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    [rJ_remove_feature,cJ_remove_feature]=size(J_remove_feature);
    if(cJ_remove_feature>1)
        if (J_remove_feature(2,1)<J_remove_feature(2,2));
            J_remove_feature=J_remove_feature;
        else
            iden_J_remove_feature=diag(ones(cJ_remove_feature,1));
            iden_J_remove_feature=rot90(iden_J_remove_feature);
            J_remove_feature=J_remove_feature*iden_J_remove_feature ;
        end
```

Gambar 5.9 Lanjutan kode program pemilihan *node* turunan dari *current node* algoritma IBB

```
64      else
65          J_remove_feature=J_remove_feature;
66      end
67      J_remove_feature=J_remove_feature';
68      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
69      % choose qk feature
70      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
71      %for k=0
72      if k==0
73          QJ(:,:,dbar+1)=J_remove_feature([1:q(dbar+1,1)],:);
74      %another
75      else
76          [row,col]=size(J_remove_feature([1:q(k,1)],:));
77          QJ(1:row,1:col,k)=J_remove_feature([1:q(k,1)],:);
78      end
79      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
80      % to avoid feature duplicate testing
81      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
82      %for k=0
83      if k==0
84          remove_fitur(dbar+1,:)=(QJ(:,1,dbar+1))';
85          nilai_J(dbar+1,:)=(QJ(:,2,dbar+1))';
86          PSI=setdiff(PSI,remove_fitur(dbar+1,:));
87          r=r-q(dbar+1);
88      %another
89      else
90          remove_fitur(k,:)=(QJ(:,1,k))';
91          nilai_J(k,:)=(QJ(:,2,k))';
92          PSI=setdiff(PSI,remove_fitur(k,:));
93          r=r-q(k);
94      end
95      J_remove_feature=[];
96      step=2;
```

Gambar 5.10 Lanjutan kode program pemilihan *node* turunan dari *current node* algoritma IBB

5.3.3 Pemeriksaan N*ode* Turunan Paling Kanan

Pada langkah ini nilai kriteria pada *level* ini akan diperiksa apakah lebih kecil dari nilai *bound* atau tidak. Jika nilai kriteria lebih kecil dari nilai *bound* saat itu, maka akan dilanjutkan pada langkah ketiga yang menunjukkan bahwa node tersebut mengalami *cut off*. Jika nilai kriteria lebih besar dari nilai *bound* saat itu dan tidak pada *level* terakhir, maka penelusuran akan dilanjutkan ke *level* selanjutnya sesuai

dengan persamaan (2.19) dan fitur *candidate* diset sesuai dengan persamaan (2.18), yaitu:

$$\overline{X_{k+1}} = \overline{X_k} \setminus \{Q_{k,q_k}\} \quad \text{.....(2.18)}$$

$$k = k+1 \quad \text{.....(2.19).}$$

Pseudocode program pemeriksaan *node* turunan paling kanan dapat dilihat pada Gambar 5.11.

5.3.4 *Cut-off*

Pada langkah ini terjadi *cut off* bila nilai kriteria lebih kecil dari *bound*. Fitur yang telah dihilangkan akan dikembalikan lagi pada himpunan kontrol fitur sesuai dengan persamaan (2.20), (2.21), (2.22) dan (2.23), yaitu:

$$\Psi = \Psi \cup \{Q_{k,q_k}\} \quad \text{.....(2.20)}$$

dan

$$r = r+1 \quad \text{.....(2.21),}$$

$$Q_k = Q_k \setminus \{Q_{k,q_k}\} \quad \text{.....(2.22)}$$

dan

$$q_k = q_k - 1 \quad \text{.....(2.23).}$$

Pseudocode program *cut off* pada *Improved Branch and Bound* ini dapat dilihat pada Gambar 5.12.

5.3.5 *Backtrack* ke *Level* Sebelumnya

Pada langkah ini dilakukan penelusuran balik ke *level tree* yang lebih rendah sesuai dengan persamaan (2.24), yaitu:

$$k = k-1 \quad \text{.....(2.24)}$$

dapat dilihat pada *pseudocode* baris ke 5 dan fitur yang telah dihilangkan akan

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                   step 2                      %
%      test the right-most descendants node     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%for k=0
if k==0
    pointer=q(dbar+1);
    if q(dbar+1)==0
        step=4;
    elseif nilai_J(dbar+1,pointer)<=bound
        step=3;
    else
        tempxbar=setdiff(xbar(dbar+1,:),remove_fitur(dbar+1,pointer));
        [rtempxbar,ctempxbar]=size(tempxbar);
        xbar(k+1,[1:ctempxbar])=tempxbar;
        if k+1==dbar
            step=5;
        else
            k=k+1;
            step=1;
        end
    end
%another
else
    pointer=q(k);
    if q(k)==0
        step=4;
    elseif nilai_J(k,pointer)<=bound
        step=3;
    else
        currentxbar=(nonzeros(xbar(k,:)))';
        [rcxbar,ccxbar]=size(currentxbar);
        xbar(k+1,[1:ccxbar-1])=setdiff(currentxbar,remove_fitur(k,pointer));
        if k+1==dbar
            step=5;
        else
            k=k+1;
            step=1;
        end
    end
end
```

Gambar 5.11 Kode program pemeriksaan *node* turunan paling kanan algoritma IBB

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                  step 3                     %
%                  cut off                    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if k==0
    PSI=[PSI,QJ(pointer,1,dbar+1)];
    r=r+1    ;
    QJ(pointer,:,dbar+1)=zeros(size(QJ(pointer,:,dbar+1)));
    q(dbar+1)=q(dbar+1)-1;
else
    PSI=[PSI,QJ(pointer,1,k)];
    r=r+1              ;
    QJ(pointer,:,k)=zeros(size(QJ(pointer,:,k)));
    q(k)=q(k)-1;
end
step=2;
```

Gambar 5.12 Kode program *cut-off* algoritma IBB

dikembalikan pada himpunan fitur *candidate* sesuai dengan persamaan (2.25), yaitu:

$$\overline{X_k} = \overline{X_{k+1}} \cup \{Q_{k,q_k}\} \quad \text{.....(2.25)}$$

dapat dilihat pada *pseudocode* baris ke 10 sampai 20. Jika *level tree* bernilai -1, maka algoritma berhenti melakukan penelusuran. *Pseudocode* program *backtrack* ke *level* sebelumnya pada algoritma *Improved Branch and Bound* ini dapat dilihat pada Gambar 5.13.

5.3.6 *Update* nilai *Bound*

Pada langkah ini nilai *bound* di-*update* dengan nilai kriteria pada *level tree* tersebut sesuai dengan persamaan (2.26), yaitu:

$$x^* = J_{k,q_k} \quad \text{.....(2.26)}$$

dapat dilihat pada *pseudocode* baris ke 6 dan 8, dan fitur terbaik saat itu disimpan sesuai dengan persamaan (2.27), yaitu:

$$X = \overline{X_{k+1}} \qquad \text{.......(2.27)}$$

dapat dilihat pada *pseudocode* baris ke 10. *Pseudocode* program *update* nilai *bound* pada algoritma *Improved Branch and Bound* ini dapat dilihat pada Gambar 5.14.

```
1   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2   %                 step 4                       %
3   %              backtracking                    %
4   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
5   k=k-1;
6   if k==-1
7       done=0;
8       break;
9   else
10      if k==0
11          pointer=q(dbar+1);
            tempxbar=sort([(nonzeros(xbar(k+1,:)))'
12  QJ(pointer,1,dbar+1)]);
13          [rtx,ctx]=size(tempxbar);
14          xbar(dbar+1,[1:ctx])=tempxbar;
15      else
16          pointer=q(k);
17          tempxbar=sort([(nonzeros(xbar(k+1,:)))' QJ(pointer,1,k)]);
18          [rtx,ctx]=size(tempxbar);
19          xbar(k,[1:ctx])=tempxbar;
20      end
21      step=3;
22  end
```

Gambar 5.13 Kode program *backtrack* ke *level* sebelumnya algoritma IBB

```
1   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2   %                 step 5                       %
3   %              update bound                    %
4   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
5   if k==0
6       bound=nilai_J(dbar+1,pointer);
7   else
8       bound=nilai_J(k,pointer);
9   end
10  x=(nonzeros(xbar(k+1,:)))';
11  step=2;
12   end
```

Gambar 5.14 Kode program *update* nilai *bound* algoritma IBB

5.4 ALGORITMA *BRANCH AND BOUND PARTIAL PREDICTION*

Terdapat enam proses pada algoritma *Branch and Bound Partial Prediction*, yaitu:

1. Inisialisasi
2. Pemilihan *node* turunan dari *current node* dengan prediksi
3. Pemeriksaan *node* turunan paling kanan
4. *Cut off*
5. *Backtrack* ke *level* sebelumnya
6. *Update* nilai *bound*

Selanjutnya akan diberikan *pseudocode* untuk setiap proses diatas.

5.4.1 Inisialisasi

Pada langkah ini variabel-variabel yang harus diinisialisasikan lebih dulu yaitu variabel *level* (k), fitur *candidate* awal $(\bar{x})$, himpunan kontrol fitur (Ψ), jumlah dari elemen himpunan kontrol fitur (r) dan *bound* (x^*). Variabel *level* diinisialisasikan dengan nol (0). Variabel fitur *candidate* awal diinisialisasikan dengan himpunan fitur awal, sebab pada *level* nol ini belum ada fitur yang dihilangkan. Variabel himpunan kontrol fitur diinisialisasikan dengan himpunan fitur awal. Variabel jumlah dari elemen himpunan kontrol fitur diinisialisasikan dengan jumlah elemen himpunan fitur awal dan variabel *bound* diinisialisasikan dengan nilai paling kecil yang mungkin. *Pseudocode* untuk proses inisialisasi dapat dilihat pada Gambar 5.15.

5.4.2 Pemilihan *Node* Turunan dari *Current Node* dengan Prediksi

Pada langkah ini didapatkan fitur-fitur yang bisa dijadikan sebagai fitur *candidate*. Pada langkah ini dilakukan suatu prediksi sederhana untuk menentukan fitur-fitur

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                  Initialization              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
D=5; % number of set fitur
d=2; % number of subset fitur
dbar=D-d; % number of remove fitur
k=0;         % level tree
r=D;
Y=[1:D];
xbar(dbar+1,:)=Y;
PSI=Y;
A=zeros(D,1);
S=zeros(D,1);
done=1;
step=1;
evaluasi=0;
s=1;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%first bound initialization
boundawal=PSI(1:d);
tempbound=nilai_kriteria4(boundawal);
bound=tempbound;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Gambar 5.15 Kode program inisialisasi algoritma BBPP

yang dijadikan sebagai fitur *candidate*. Jumlah dari banyaknya fitur *candidate* didapatkan dengan rumus persamaan (3.5), yaitu:

$$q_k = r - (D - d - k - 1) \quad \text{..} (3.5).$$

Fitur-fitur diurutkan secara *descending* didasarkan pada nilai prediksi tiap fitur yang telah diperoleh sesuai dengan persamaan (3.6), yaitu:

$$A_{\psi_{j1}} \ge A_{\psi_{j2}} \ge \ldots \ge A_{\psi_{jr}} \quad \text{..} (3.6).$$

Hal ini berbeda dengan algoritma *Improved Branch and Bound* yang mengurutkan fitur secara *ascending* berdasarkan pada nilai kriteria sebenarnya. Setelah fitur-

fitur diurutkan, selanjutnya dipilih jumlah fitur yang diminta dan nilai kriteria sebenarnya dari fitur yang dipilih dihitung sesuai dengan persamaan (3.7) dan (3.8), yaitu:

$$Q_{k,i} = \psi_{j_i} \qquad (3.7)$$

untuk i = 1,...,q_k

$$J_{k,i} = J\left(\overline{X_k} \setminus \{\psi_{j_i}\}\right) \qquad (3.8)$$

untuk i = 1,...,q_k.

Fitur-fitur yang telah dijadikan fitur *candidate* dihilangkan dari himpunan kontrol fitur sesuai dengan persamaan (3.9) dan (3.10), yaitu:

$$\Psi = \Psi \setminus Q_k \qquad (3.9)$$

dan

$$r = r - q_k \qquad (3.10).$$

Pseudocode untuk proses pemilihan *node* turunan dari *current node* dapat dilihat pada Gambar 5.16, 5.17 dan 5.18.

5.4.3 Pemeriksaan *Node* Turunan Paling Kanan

Pada langkah ini nilai kriteria pada *level* ini akan diperiksa apakah lebih kecil dari nilai *bound* atau tidak. Jika nilai nilai kriteria lebih kecil dari nilai *bound* saat itu, maka akan dilanjutkan pada langkah ketiga yang menunjukkan bahwa node tersebut mengalami *cut off*. Jika nilai kriteria lebih besar dari nilai *bound* saat itu dan tidak pada *level* terakhir, maka penelusuran akan dilanjutkan ke *level* selanjutnya sesuai dengan persamaan (3.12), yaitu:

$$k = k + 1 \qquad (3.12)$$

dan fitur *candidate* diset sesuai dengan persamaan (3.11), yaitu:

$$\overline{X_{k+1}} = \overline{X_k} \setminus \{Q_{k,q_k}\} \quad \text{...(3.11).}$$

Pseudocode program pemeriksaan *node* turunan paling kanan dapat dilihat pada Gambar 5.19.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                      step 1                                %
%select descendants of the current node with prediction     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%for k=0
if k==0
    q(dbar+1,1)=r-(D-d-k-1);
    for j=1:r
        psi=PSI(1,j);
        %%%%%%%%%%%%%%%%%%%%%%%
        % compute A
        %%%%%%%%%%%%%%%%%%%%%%
        Jsatu=nilai_kriteria4(xbar(dbar+1,:));
        Jdua=nilai_kriteria4(setdiff(xbar(dbar+1,:),psi));
        A(j,1)=(A(j,1)*S(j,1)+Jsatu-Jdua)/(S(j,1)+1);
        S(j,1)=S(j,1)+1  ;
        FiturPrediksi(j,:)=[psi A(j,1)];
        Jremfitur(j,:)=[psi Jdua];
        psi=[];
    end
%another
else
    q(k,1)=r-(D-d-k-1);
    for j=1:r
        psi=PSI(1,j);
        %%%%%%%%%%%%%%%%%%%%%%%
        %compute A
        %%%%%%%%%%%%%%%%%%%%%%
        Jsatu=nilai_kriteria4((nonzeros(xbar(k,:)))');;
Jdua=nilai_kriteria4(setdiff((nonzeros(xbar(k,:)))',psi));
        A(j,1)=(A(j,1)*S(j,1)+Jsatu-Jdua)/(S(j,1)+1);
        S(j,1)=S(j,1)+1  ;
        FiturPrediksi(j,:)=[psi A(j,1)];
        Jremfitur(j,:)=[psi Jdua];
        psi=[];
    end
end
```

Gambar 5.16 Kode program pemilihan *node* turunan dari *current node* algoritma BBPP

```
FiturPrediksi=FiturPrediksi';
Jremfitur=Jremfitur';
[rFP,cFP]=size(FiturPrediksi);
if(cFP>1)
    if (FiturPrediksi(2,1)>FiturPrediksi(2,2));
        FiturPrediksi=FiturPrediksi;
        Jremfitur=Jremfitur;
    else
        iden_FiturPrediksi=diag(ones(cFP,1));
        iden_FiturPrediksi=rot90(iden_FiturPrediksi);
        FiturPrediksi=FiturPrediksi*iden_FiturPrediksi ;
        Jremfitur=Jremfitur*iden_FiturPrediksi;
    end
else
    FiturPrediksi=FiturPrediksi;
    Jremfitur=Jremfitur;
end
FiturPrediksi=FiturPrediksi';
Jremfitur=Jremfitur';
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% choose qk feature
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%for k=0
if k==0
    for a=1:q(dbar+1)
        evaluasi=evaluasi+1;
    end
    QJ(:,:,dbar+1)=Jremfitur([1:q(dbar+1,1)],:);
    matrikevaluasi(s,1)=[evaluasi];
    evaluasi=0;
    s=s+1;
%another
else
    for a=1:q(k)
        evaluasi=evaluasi+1;
    end
    matrikevaluasi(s,1)=[evaluasi];
    evaluasi=0;
    s=s+1;
    [row,col]=size(Jremfitur([1:q(k,1)],:));
    QJ(1:row,1:col,k)=Jremfitur([1:q(k,1)],:);
end
%for k=0
if k==0
    fiturhilang(dbar+1,:)=(QJ(:,1,dbar+1))';
    nilai_J(dbar+1,:)=(QJ(:,2,dbar+1))';
```

Gambar 5.17 Lanjutan kode program pemilihan *node* turunan dari *current node* algoritma BBPP

```
85      PSI=setdiff(PSI,fiturhilang(dbar+1,:));
86      r=r-q(dbar+1);
87  %another
88  else
89      fiturhilang(k,:)=(QJ(:,1,k))';
90      nilai_J(k,:)=(QJ(:,2,k))';
91      PSI=setdiff(PSI,fiturhilang(k,:));
92      r=r-q(k);
93  end
94  FiturPrediksi=[];
95  Jremfitur=[];
96  step=2;
```

Gambar 5.18 Lanjutan kode program pemilihan *node* turunan dari *current node* algoritma BBPP

5.4.4 *Cut-off*

Pada langkah ini terjadi *cut-off* bila nilai kriteria lebih kecil dari *bound*. Fitur yang telah dihilangkan akan dikembalikan lagi pada himpunan kontrol fitur sesuai dengan persamaan :

$$\Psi = \Psi \cup \{Q_{k,q_k}\} \quad \ldots\ldots(3.13)$$

dan

$$r = r + 1 \quad \ldots\ldots(3.14),$$

$$Q_k = Q_k \setminus \{Q_{k,q_k}\} \quad \ldots\ldots(3.15)$$

dan

$$q_k = q_k - 1 \quad \ldots\ldots(3.16).$$

Pseudocode program *cut off* pada algoritma BBPP ini dapat dilihat pada Gambar 5.20.

5.4.5 *Backtrack* ke *Level* Sebelumnya

Pada langkah ini dilakukan penelusuran balik ke *level tree* yang lebih rendah sesuai dengan persamaan (3.17), yaitu:

$$k = k - 1 \quad \text{...(3.17)}$$

dan fitur yang telah dihilangkan akan dikembalikan pada himpunan fitur *candidate* sesuai dengan persamaan (3.18), yaitu:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                  step 2                      %
%     test the right-most descendants node     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if k==0
    pointer=q(dbar+1);
    if q(dbar+1)==0
        step=4;
    elseif nilai_J(dbar+1,pointer)<=bound
        step=3;
    else
        tempxbar=setdiff(xbar(dbar+1,:),fiturhilang(dbar+1,pointer));
        [rtempxbar,ctempxbar]=size(tempxbar);
        xbar(k+1,[1:ctempxbar])=tempxbar;
        if k+1==dbar
            step=5;
        else
            k=k+1;
            step=1;
        end
    end
else
    pointer=q(k);
    if q(k)==0
        step=4;
    elseif nilai_J(k,pointer)<=bound
         step=3;
    else
         currentxbar=(nonzeros(xbar(k,:)))';
         [rcxbar,ccxbar]=size(currentxbar);
         xbar(k+1,[1:ccxbar-1])=setdiff(currentxbar,fiturhilang(k,pointer));
         if k+1==dbar
            step=5;
         else
            k=k+1;
            step=1;
         end
    end
End
```

Gambar 5.19 Kode program pemeriksaan *node* turunan paling kanan algoritma BBPP

$$\overline{X_k} = \overline{X_{k+1}} \cup \{Q_{k,q_k}\} \quad \text{.........(3.18).}$$

Jika *level tree* bernilai -1, maka algoritma berhenti melakukan penelusuran. *Pseudocode* program *backtrack* ke *level* sebelumnya pada *Improved Branch and Bound* ini dapat dilihat pada Gambar 5.21.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                    step 3                     %
%                   cut off                     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if k==0
    PSI=[PSI,QJ(pointer,1,dbar+1)];
    r=r+1;
    QJ(pointer,:,dbar+1)=zeros(size(QJ(pointer,:,dbar+1)));
    q(dbar+1)=q(dbar+1)-1;
else
    PSI=[PSI,QJ(pointer,1,k)];
    r=r+1;
    QJ(pointer,:,k)=zeros(size(QJ(pointer,:,k)));
    q(k)=q(k)-1;
end
step=2;
```

Gambar 5.20 Kode program *cut-off* algoritma BBPP

5.4.6 *Update* nilai *Bound*

Pada langkah ini nilai *bound* di-*update* dengan nilai kriteria pada *level tree* tersebut sesuai dengan persamaan (3.19), yaitu:

$$x^* = J_{k,q_k} \quad \text{.........(3.19)}$$

dan fitur terbaik saat itu disimpan sesuai dengan persamaan (3.20), yaitu:

$$X = \overline{X_{k+1}} \quad \text{.........(3.20).}$$

Pseudocode program *update* nilai *bound* pada algoritma BBPP ini dapat dilihat pada Gambar 5.22.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                       step 4                   %
%                    backtracking                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
k=k-1;
if k==-1
    done=0;
    break;
else
    if k==0
        pointer=q(dbar+1);
        tempxbar=sort([(nonzeros(xbar(k+1,:)))' QJ(pointer,1,dbar+1)]);
        [rtx,ctx]=size(tempxbar);
        xbar(dbar+1,[1:ctx])=tempxbar;
    else
        pointer=q(k);
        tempxbar=sort([(nonzeros(xbar(k+1,:)))' QJ(pointer,1,k)]);
        [rtx,ctx]=size(tempxbar);
        xbar(k,[1:ctx])=tempxbar;
    end
    step=3;
end
```

Gambar 5.21 Kode program *backtrack* ke *level* sebelumnya algoritma BBPP

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                       step 5                   %
%                    update bound                %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if k==0
    bound=nilai_J(dbar+1,pointer);
else
    bound=nilai_J(k,pointer);
end
x=(nonzeros(xbar(k+1,:)))';
step=2;
```

Gambar 5.22 Kode program *update* nilai *bound* algoritma BBPP

5.5 ALGORITMA *FAST BRANCH AND BOUND*

Terdapat enam proses pada algoritma *Fast Branch and Bound*, yaitu:

1. Inisialisasi
2. Pemilihan *node* turunan dari *current node*
3. Pemeriksaan *node* turunan paling kanan

4. *Cut off*
5. *Backtrack* ke *level* sebelumnya
6. *Update* nilai *bound*

Selanjutnya akan diberikan *pseudocode* untuk setiap proses diatas.

5.5.1 Inisialisasi

Pada langkah ini variabel-variabel yang harus diinisialisasikan lebih dulu yaitu variabel *level* (k), fitur *candidate* awal $(\bar{x})$, himpunan kontrol fitur (Ψ), jumlah dari elemen himpunan kontrol fitur (r) dan *bound* (x^*). Variabel *level* diinisialisasikan dengan nol (0). Variabel fitur *candidate* awal diinisialisasikan dengan himpunan fitur awal, sebab pada *level* nol ini belum ada fitur yang dihilangkan. Variabel himpunan kontrol fitur diinisialisasikan dengan himpunan fitur awal. Variabel jumlah dari elemen himpunan kontrol fitur diinisialisasikan dengan jumlah elemen himpunan fitur awal dan variabel *bound* diinisialisasikan dengan nol (0). *Pseudocode* untuk proses inisialisasi dapat dilihat pada Gambar 5.23.

5.5.2 Pemilihan *Node* Turunan dari *Current Node*

Pada langkah ini didapatkan fitur-fitur yang bisa dijadikan sebagai fitur *candidate*. Jumlah dari banyaknya fitur *candidate* didapatkan dengan rumus persamaan (3.23), yaitu:

$$q_k = r - (D - d - k - 1) \quad \text{...(3.23)}$$

dapat dilihat pada *pseudocode* baris ke 5 dan 23. Fitur-fitur diurutkan secara *ascending* sesuai dengan persamaan (3.26) dimana nilainya didapatkan sesuai dengan persamaan (3.24) dan (3.25), yaitu:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                  Initialization              %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
DD=5; % number of set fitur
d=2; % number of subset fitur
dbar=D-d; % number of remove fitur
k=0;         % level tree
r=D;
Y=[1:D];
xbar(dbar+1,:)=Y;
PSI=Y;
A=zeros(D,1);
S=zeros(D,1);
done=1;
step=1;
criterion_evaluation=0
bound=0;
delta=1;
gama=1;
```

Gambar 5.23 Kode program inisialisasi algoritma FBB

$$v_j = J_{k-1,q_{k-1}} - A_{\psi_j} \quad \text{(3.24)}$$

$$v_j = J\left(\overline{x_k} \setminus \{\psi_j\}\right) \quad \text{(3.25)}$$

$$v_{j_1} \le v_{j_2} \le \ldots \le v_{j_r} \quad \text{(3.26)}$$

dapat dilihat pada *pseudocode* baris 6 sampai 53. Setelah fitur-fitur diurutkan, selanjutnya dipilih jumlah fitur yang diminta dan nilai kriteria sebenarnya dari fitur yang dipilih dihitung sesuai dengan persamaan (3.27), (3.28), (3.29), (3.30) dan (3.31), yaitu:

$$Q_{k,i} = \psi_{j_i} \quad \text{(3.27)}$$

dan

$$J_{k,i} = v_{j_i} \quad \text{(3.28)},$$

jika v_{j_i} merupakan sebuah rekord yang nilainya dihitung dan

$$J_{k,i} = J_{k-1,q_{k-1}} - \gamma . A_{\psi_{j_i}} \quad \text{(3.29)},$$

jika v_{j_i} merupakan sebuah rekord yang nilainya diprediksi.

$T_{k,i} =$"C" .....(3.30),

jika v_{j_i} merupakan sebuah rekord yang nilainya dihitung dan

$T_{k,i} =$"P" .....(3.31)

jika v_{j_i} merupakan sebuah rekord yang nilainya diprediksi. Implementasi dari persamaan ini dapat dilihat pada *pseudocode* baris ke 58 sampai 79. Fitur-fitur yang telah dijadikan fitur *candidate* dihilangkan dari himpunan kontrol fitur sesuai dengan persamaan (3.32) dan (3.33), yaitu:

$\Psi = \Psi \setminus Q_k$ .....(3.32)

dan

$r = r - q_k$ .....(3.33)

dapat dilihat pada *pseudocode* baris ke 83 sampai 104. *Pseudocode* untuk proses pemilihan *node* turunan dari *current node* dapat dilihat pada Gambar 5.24,5.25 dan 5.26.

5.5.3 Pemeriksaan *Node* Turunan Paling Kanan

Pada langkah ini akan diperiksa apakah nilai kriteria lebih kecil dari *bound* dan vektor tipenya "P", maka nilai kriteria sebenarnya harus dihitung sesuai dengan persamaan (3.34) dan (3.35), yaitu:

$J_{k,q_k} = J(\overline{x_k} \setminus \{Q_{k,q_k}\})$ .....(3.34)

dan

$T_{k,q_k} =$"C" .....(3.35)

dapat dilihat pada *pseudocode* baris 9 sampai 14. Jika nilai kriterianya lebih kecil dari *bound* dan vektor tipenya "C", maka akan dilanjutkan ke langkah ketiga

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                     step 1                     %
%    select descendants of the current node      %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    q(dbar+1,1)=r-(D-d-k-1);
    for j=1:r
        psi=PSI(1,j);
        if (k+1<D-d)&(S(psi,1)>=delta)
            %prediction allowed --> prediction flag=2
            J(dbar+1,:)=nilai_kriteria4(xbar(dbar+1,:));
            v(j,1)=J(dbar+1,:)-A(j,1);
            T(j,dbar+1)=[2];
        else
            %true criterion value computed-->computed flag=1
v(j,1)=nilai_kriteria4((nonzeros(setdiff(xbar(dbar+1,:),psi)))');
            %criterion_evaluation=criterion_evaluation+1
            T(j,dbar+1)=[1];
        end
            Jremfitur(j,:)=[psi v(j,1) T(j,dbar+1) A(j,1) S(j,1)];
            psi=[];
    end
else
    q(k,1)=r-(D-d-k-1);
    for j=1:r
        psi=PSI(1,j);
        if (k+1<D-d)&(S(psi,1)>=delta)
            %prediction allowed
            J(k,:)=nilai_kriteria4(xbar(k,:));
            v(j,1)=J(k,:)-A(j,1);
            T(j,k)=[2];
        else
            %true criterion value computed
v(j,1)=nilai_kriteria4((nonzeros(setdiff(xbar(k,:),psi)))');
            %criterion_evaluation=criterion_evaluation+1
            T(j,k)=[1];
        end
        Jremfitur(j,:)=[psi v(j,1) T(j,k) A(j,1) S(j,1)];
        psi=[];
    end
end
```

Gambar 5.24 Kode program pemilihan *node* turunan dari *current node* algoritma FBB

```
Jremfitur=Jremfitur';
[rJremfitur,cJremfitur]=size(Jremfitur);
if(cJremfitur>1)
    if (Jremfitur(2,1)<Jremfitur(2,2))
        Jremfitur=Jremfitur;
    else
        iden_Jremfitur=diag(ones(cJremfitur,1));
        iden_Jremfitur=rot90(iden_Jremfitur);
        Jremfitur=Jremfitur*iden_Jremfitur;
    end
else
    Jremfitur=Jremfitur;
end
Jremfitur=Jremfitur';
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% choose qk feature
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if k==0
    for i=1:q(dbar+1)
        if (Jremfitur(i,3)==1)
            Jremfitur(i,2)=Jremfitur(i,2);
            criterion_evaluation=criterion_evaluation+1
        else
            Jremfitur(i,2)=(nilai_kriteria4(xbar(dbar+1,:)))-
gama*Jremfitur(i,4);
        end
    end
    QJ(:,:,dbar+1)=Jremfitur([1:q(dbar+1,1)],:);
else
    for i=1:q(k)
        if (Jremfitur(i,3)==1)
            Jremfitur(i,2)=Jremfitur(i,2);
            criterion_evaluation=criterion_evaluation+1
        else
            Jremfitur(i,2)=(nilai_kriteria4(xbar(k,:)))-
(Jremfitur(i,4));
        end
    end
    [row,col]=size(Jremfitur([1:q(k,1)],:));
    QJ(1:row,1:col,k)=Jremfitur([1:q(k,1)],:);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% to avoid feature duplicate testing
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if k==0
    fiturhilang(dbar+1,:)=(QJ(:,1,dbar+1))';
    nilai_J(dbar+1,:)=(QJ(:,2,dbar+1))';
    nilai_T(dbar+1,:)=(QJ(:,3,dbar+1))';
```

Gambar 5.25 Lanjutan kode program pemilihan *node* turunan dari *current node* algoritma FBB

```
87        nilai_A(dbar+1,:)=(QJ(:,4,dbar+1))';
88        nilai_S(dbar+1,:)=(QJ(:,5,dbar+1))';
89        PSI=setdiff(PSI,fiturhilang(dbar+1,:));
90        r=r-q(dbar+1);
91    else
92        prak=k-1;
93        if prak==0
94            prak=dbar+1;
95        end
96        [rT,cT]=size((QJ(:,3,k))');
97        fiturhilang(k,:)=(QJ(:,1,k))';
98        nilai_J(k,:)=(QJ(:,2,k))';
99        nilai_T(k,1:cT)=(QJ(:,3,k))';
100       nilai_A(k,:)=(QJ(:,4,prak))';
101       nilai_S(k,:)=(QJ(:,5,prak))';
102       PSI=setdiff(PSI,fiturhilang(k,:));
103       r=r-q(k);
104   end
105   Jremfitur=[];
106   v=[];
107   step=2;
```

Gambar 5.26 Lanjutan kode program pemilihan *node* turunan dari *current node* algoritma FBB

untuk dilakukan *cut off* pada *node* tersebut. Jika nilai kriteria lebih besar dari nilai *bound* saat itu dan tidak pada *level* terakhir, maka penelusuran akan dilanjutkan ke *level* selanjutnya sesuai dengan persamaan (3.37), yaitu:

$$k = k+1 \quad \text{.....(3.37)}$$

dapat dilihat pada *pseudocode* baris ke 25 dan fitur *candidate* diset sesuai dengan persamaan (3.36), yaitu:

$$\overline{X_{k+1}} = \overline{X_k} \setminus \{Q_{k,q_k}\} \quad \text{.....(3.36)}$$

dapat dilihat pada *pseudocode* baris ke 19 sampai 21. *Pseudocode* program pemeriksaan *node* turunan paling kanan dapat dilihat pada Gambar 5.27, 5.28, dan 5.29.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                   step 2                      %
%       test the right-most descendants node    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if k==0
    pointer=q(dbar+1);
    if q(dbar+1)==0
        step=4;
    elseif (nilai_T(dbar+1,pointer)==2)&(nilai_J(dbar+1,pointer)<=bound)
        %true criterion value computed
nilai_J(dbar+1,pointer)=nilai_kriteria4((nonzeros(setdiff(xbar(dbar+1,:),
                              fiturhilang(dbar+1,pointer))))');
        T(dbar+1,pointer)=[1];
        nilai_T(dbar+1,pointer)=[1];
        QJ(pointer,3,dbar+1)=[1];
        criterion_evaluation=criterion_evaluation+1
        if (nilai_T(dbar+1,pointer)==1)&(nilai_J(dbar+1,pointer)<=bound)
            step=3;
        else
            tempxbar=setdiff((nonzeros(xbar(dbar+1,:)))',
                              fiturhilang(dbar+1,pointer));
            [rtempxbar,ctempxbar]=size(tempxbar);
            xbar(k+1,[1:ctempxbar])=tempxbar;
            if k+1==dbar
                step=7;
            else
                k=k+1;
                prevk=k-1;
                if prevk==0
                    Tparrent=nilai_T(dbar+1,q(dbar+1));
                else
                    Tparrent=nilai_T(prevk,q(prevk));
                end
                Tnode=nilai_T(dbar+1,pointer);
                if Tparrent==1 & Tnode==1
                    step=6;
                else
                    step=1;
                end
            end
        end
    elseif (nilai_T(dbar+1,pointer)==1)&(nilai_J(dbar+1,pointer)<=bound)
        step=3;
    else
tempxbar=setdiff((nonzeros(xbar(dbar+1,:)))',fiturhilang(dbar+1,pointer));
        [rtempxbar,ctempxbar]=size(tempxbar);
        xbar(k+1,[1:ctempxbar])=tempxbar;
```

Gambar 5.27 Kode program pemeriksaan *node* turunan paling kanan algoritma FBB

```
46          if k+1==dbar
47              step=7;
48          else
49              k=k+1;
50              prevk=k-1;
51              if prevk==0
52                  Tparrent=nilai_T(dbar+1,q(dbar+1));
53              else
54                  Tparrent=nilai_T(prevk,q(prevk));
55              end
56              Tnode=nilai_T(dbar+1,pointer);
57              if Tparrent==1 & Tnode==1
58                  step=6;
59              else
60                  step=1;
61              end
62          end
63      end
64  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
65  else
66      pointer=q(k);
67      if q(k)==0
68          step=4;
69      elseif (nilai_T(k,pointer)==2)&(nilai_J(k,pointer)<=bound)
70          %true criterion value computed
71          nilai_J(k,pointer)=nilai_kriteria4((nonzeros(setdiff(xbar(k,:),
                        fiturhilang(k,pointer))))');
72          T(k,pointer)=[1];
73          nilai_T(k,pointer)=[1];
74          criterion_evaluation=criterion_evaluation+1
75          if (nilai_T(k,pointer)==1)&(nilai_J(k,pointer)<=bound)
76              step=3;
77          else
78              tempxbar=setdiff((nonzeros(xbar(k,:)))',fiturhilang(k,pointer));
79              [rtempxbar,ctempxbar]=size(tempxbar);
80              xbar(k+1,[1:ctempxbar])=tempxbar;
81              if k+1==dbar
82                  step=7;
83              else
84                  k=k+1;
85                  if k==0
86                      step=1;
87                  else
88                      prevk=k-1;
89                      if prevk==0
90                          Tparrent=nilai_T(dbar+1,q(dbar+1));
91                      else
92                          Tparrent=nilai_T(prevk,q(prevk));
93                      end
```


Gambar 5.28 Lanjutan kode program pemeriksaan *node* turunan paling kanan algoritma FBB

```
94                      Tnode=nilai_T(prevk,pointer);
95                      if Tparrent==1 & Tnode==1
96                          step=6;
97                      else
98                          step=1;
99                      end
100                 end
101             end
102         end
103         elseif (nilai_T(k,pointer)==1)&(nilai_J(k,pointer)<=bound)
104             step=3;
105         else
106             tempxbar=setdiff((nonzeros(xbar(k,:)))',fiturhilang(k,pointer));
107             [rtempxbar,ctempxbar]=size(tempxbar);
108             xbar(k+1,[1:ctempxbar])=tempxbar;
109             if k+1==dbar
110                 step=7;
111             else
112             k=k+1;
113             if k==0
114                 step=1;
115             else
116                 prevk=k-1;
117                 if prevk==0
118                    Tparrent=nilai_T(dbar+1,q(dbar+1));
119                 else
120                    Tparrent=nilai_T(prevk,q(prevk));
121                 end
122                 Tnode=nilai_T(prevk,pointer);
123                 if Tparrent==1 & Tnode==1
124                     step=6;
125                 else
126                     step=1;
127                 end
128             end
129         end
130     end
131 end
```

Gambar 5.29 Lanjutan kode program pemeriksaan *node* turunan paling kanan algoritma FBB

5.5.4 *Cut off*

Pada langkah ini terjadi *cut off* bila nilai kriteria lebih kecil dari *bound*. Fitur yang telah dihilangkan akan dikembalikan lagi pada himpunan kontrol fitur sesuai dengan persamaan (3.38), (3.39), (3.40) dan (3.41), yaitu:

$$\Psi = \Psi \cup \{Q_{k,q_k}\} \quad \ldots\ldots(3.38)$$

dapat dilihat pada *pseudocode* baris ke 6 dan 11, dan

$r = r + 1$ ....(3.39)

dapat dilihat pada *pseudocode* baris ke 7 dan 10,

$Q_k = Q_k \setminus \{Q_{k,q_k}\}$ ....(3.40)

dapat dilihat pada *pseudocode* baris ke 8 dan 13, dan

$q_k = q_k - 1$ ....(3.41)

dapat dilihat pada *pseudocode* baris ke 9 dan 14. *Pseudocode* program *cut off* pada algoritma *Fast Branch and Bound* ini dapat dilihat pada Gambar 5.30.

```
1  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2  %                  step 3                      %
3  %                  cut off                     %
4  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
5  if k==0
6      PSI=[PSI,QJ(pointer,1,dbar+1)];
7      r=r+1;
8      QJ(pointer,1,dbar+1)=zeros(size(QJ(pointer,1,dbar+1)));
9      q(dbar+1)=q(dbar+1)-1;
10 else
11     PSI=[PSI,QJ(pointer,1,k)];
12     r=r+1;
13     QJ(pointer,1,k)=zeros(size(QJ(pointer,1,k)));
14     q(k)=q(k)-1;
15 end
16 step=2;
```

Gambar 5.30 Kode program *cut-off* algoritma FBB

5.5.5 *Backtrack* ke *Level* Sebelumnya

Pada langkah ini dilakukan penelusuran balik ke *level tree* yang lebih rendah sesuai dengan persamaan (3.42), yaitu:

$k = k - 1$ ....(3.42)

dapat dilihat pada *pseudocode* baris ke 5 dan fitur yang telah dihilangkan akan dikembalikan pada himpunan fitur *candidate* sesuai dengan persamaan (3.43), yaitu:

$$\overline{X_k} = \overline{X_{k+1}} \cup \{Q_{k,q_k}\} \quad \text{..........(3.43)}$$

dapat dilihat pada *pseudocode* baris ke 10 sampai 20. Jika *level tree* bernilai -1, maka algoritma berhenti melakukan penelusuran. *Pseudocode* program *backtrack* ke *level* sebelumnya pada algoritma *Fast Branch and Bound* ini dapat dilihat pada Gambar 5.31.

```
1  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2  %                    step 4                    %
3  %                 backtracking                 %
4  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
5  k=k-1;
6  if k==-1
7      done=0;
8      break
9  else
10     if k==0
11         pointer=q(dbar+1);
12         tempxbar=sort([(nonzeros(xbar(k+1,:)))' QJ(pointer,1,dbar+1)]);
13         [rtx,ctx]=size(tempxbar);
14         xbar(dbar+1,[1:ctx])=tempxbar;
15     else
16         pointer=q(k);
17         tempxbar=sort([(nonzeros(xbar(k+1,:)))' QJ(pointer,1,k)]);
18         [rtx,ctx]=size(tempxbar);
19         xbar(k,[1:ctx])=tempxbar;
20      end
21     step=3;
22 end
```

Gambar 5.31 Kode program *backtrack* ke *level* sebelumnya algoritma FBB

5.5.6 *Update* nilai *Bound*

Pada langkah ini nilai *bound* di-*update* dengan nilai kriteria pada *level tree* tersebut sesuai dengan persamaan (3.44), yaitu:

$$x^* = J_{k.q_k} \quad \text{..........(3.44)}$$

dapat dilihat pada *pseudocode* baris ke 6 dan 8, dan fitur terbaik saat itu disimpan sesuai dengan persamaan (3.45), yaitu:

$$X = \overline{X_{k+1}} \quad \text{...}(3.45),$$

dapat dilihat pada *pseudocode* baris ke 10. *Pseudocode* program *update* nilai *bound* pada algoritma *Fast Branch and Bound* ini dapat dilihat pada Gambar 5.32.

```
1  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2  %                        step 5                          %
3  %                     update bound                       %
4  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
5  if k==0
6      bound=nilai_J(dbar+1,pointer);
7  else
8      bound=nilai_J(k,pointer);
9  end
10 x=(nonzeros(xbar(k+1,:)))';
11 step=2;
```

Gambar 5.32 Kode program *update* nilai *bound* algoritma FBB

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                      update A                          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if k>0
    if k==1
        Jsatu=nilai_kriteria4(xbar(dbar+1,:));
        Jdua=nilai_kriteria4((nonzeros(setdiff(xbar(k,:),
                        fiturhilang(dbar+1,pointer))))');
        %criterion_evaluation=criterion_evaluation+1
        indeksA=fiturhilang(dbar+1,pointer);
        A(indeksA,1)=((A(indeksA,1)*S(indeksA,1))+
                          Jsatu-Jdua)/(S(indeksA,1)+1);
        QJ(pointer,4,dbar+1)=A(indeksA,1);
        S(indeksA,1)=S(indeksA,1)+1;
        QJ(pointer,5,dbar+1)=S(indeksA,1);
    else
        Jsatu=nilai_kriteria4(xbar(k-1,:));
        Jdua=nilai_kriteria4((nonzeros(setdiff(xbar(k,:),
                          fiturhilang(k-1,pointer))))');
        %criterion_evaluation=criterion_evaluation+1
        indeksA=fiturhilang(k-1,pointer);
        A(indeksA,1)=((A(indeksA,1)*S(indeksA,1))+
                           Jsatu-Jdua)/(S(indeksA,1)+1);
        QJ(pointer,4,k-1)=A(indeksA,1);
        S(indeksA,1)=S(indeksA,1)+1;
        QJ(pointer,5,k-1)=S(indeksA,1);
    end
```

Gambar 5.33 Kode program *update* vektor kontribusi dan vektor kounter algoritma FBB

5.5.7 *Update* Vektor Kontribusi dan Vektor Kounter

Pseudocode program *update* vektor kontribusi dan vektor kounter pada algoritma *Fast Branch and Bound* ini dapat dilihat pada Gambar 5.33.

BAB VI

UJI COBA DAN EVALUASI

Pada bab ini dijelaskan mengenai uji coba dan evaluasi yang telah dilakukan. Pembahasan yang dikemukakan meliputi lingkungan uji coba, dan uji coba dengan menggunakan data sintetik [SPK04].

6.1 LINGKUNGAN UJI COBA

Pada bagian ini akan dijelaskan mengenai lingkungan peengujian aplikasi, yang meliputi perangkat lunak dan perangkat keras yang digunakan. Spesifikasi perangkat lunak dan perangkat keras yang digunakan dalam pembangunan aplikasi optimasi pemilihan fitur ini dapat dilihat pada tabel 6.1.

Tabel 6.1 Lingkungan Pengujian Aplikasi

Perangkat Keras	Prosesor : Intel Pentium 4 2.80 GHz Memori : 1 GB
Perangkat Lunak	Sistem Operasi : Windows Perangkat Lunak Pembangun : Matlab 6.5.1

6.2 DATA UJI COBA

Data sintetik yang digunakan pada optimasi pemilihan fitur ini berasal dari paper [SPK04]. Pada data sintetik ini telah didapatkan fungsi kriteria yang digunakan. Jumlah fitur awal yang digunakan sebesar 30 fitur yang telah diurutkan berdasarkan sifat *monotonicity*. Pada uji coba ini dievaluasi seberapa

banyak evaluasi fungsi kriteria yang terjadi pada keempat algoritma *branch and bound.*

Terdapat lima jenis uji coba yang dilakukan, yaitu:

- Percobaan 1, uji coba pemilihan fitur dengan fungsi kriteria $J(\bar{x}) = \sum_{\xi_i \in x} i^2$,
- Percobaan 2, uji coba pemilihan fitur dengan fungsi kriteria $J(\bar{x}) = \sum_{\xi_i \in x} 2^{(i-1)}$,
- Percobaan 3, uji coba pemilihan fitur dengan fungsi kriteria $J(\bar{x}) = \sum_{\xi_i \in x} i^3$,
- Percobaan 4, uji coba pemilihan fitur dengan fungsi kriteria $J(\bar{x}) = \sum_{\xi_i \in x} i$,
- Percobaan 5, uji coba pemilihan fitur dengan fungsi kriteria $J(\bar{x}) = \sum_{\xi_i \in x} |\bar{x}|$,

Ujicoba yang dilakukan dengan kelima fungsi kriteria diatas diujicobakan pada keempat algoritma *branch and bound.*

6.3 UJI COBA PERTAMA

Fungsi kriteria yang digunakan adalah $J(\bar{x}) = \sum_{\xi_i \in x} i^2$. Hasil evaluasi fungsi kriteria dari keempat algoritma *branch and bound* dapat dilihat pada Gambar 6.1 dan Tabel 6.2. D merupakan jumlah fitur awal, d merupakan jumlah fiitur yang akan dipilih, evaluasi kriteria FBB merupakan jumlah evaluasi kriteria yang terjadi pada algoritma FBB, evaluasi kriteria BBPP merupakan jumlah evaluasi kriteria yang terjadi pada algoritma BBPP, evaluasi kriteria IBB merupakan

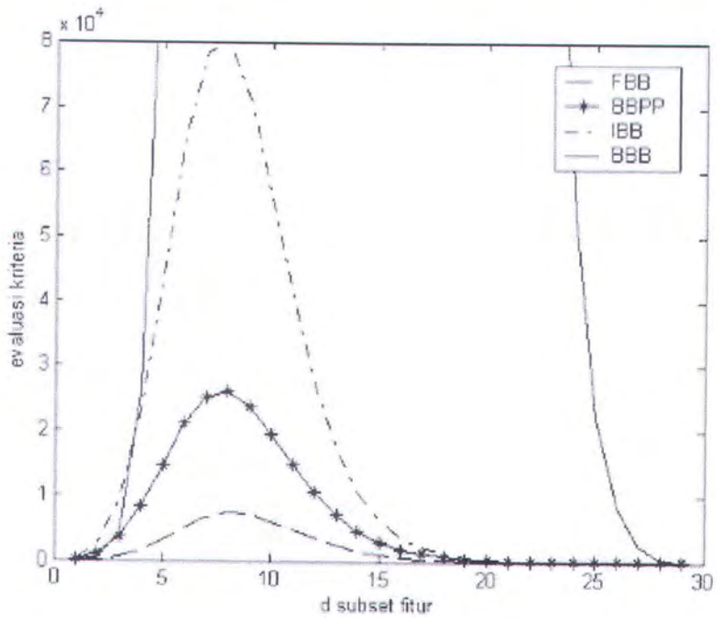
jumlah evaluasi kriteria yang terjadi pada algoritma IBB dan evaluasi kriteria BBB merupakan jumlah evaluasi kriteria yang terjadi pada algoritma BBB.

Tabel 6.2 Hasil Percobaan 1

D	d	Evaluasi kriteria FBB	Evaluasi kriteria BBPP	Evaluasi kriteria IBB	Evaluasi kriteria BBB
30	1	30	31	30	30
30	2	161	1.146	2.441	435
30	3	576	3.618	9.157	4.060
30	4	1.556	8.310	23.394	25.946
30	5	3.162	14.743	44.192	121.822
30	6	5.050	21.120	65.211	439.336
30	7	6.592	25.289	78.472	1.252.242
30	8	7.295	26.102	79.899	2.881.194
30	9	7.035	23.792	70.887	5.443.041
30	10	6.062	19.586	56.167	8.571.300
30	11	4.777	14.851	40.600	11.413.455
30	12	3.524	10.592	27.429	13.039.372
30	13	2.450	7.161	17.474	12.974.378
30	14	1.638	4.670	10.700	11.420.335
30	15	1.057	2.945	6.330	9.034.241
30	16	692	1.867	3.705	6.522.050
30	17	448	1.173	2.134	4.355.601
30	18	278	706	1.200	2.723.591
30	19	184	449	701	1.607.928
30	20	122	283	401	902.198
30	21	81	177	223	482.220
30	22	65	133	148	245.592
30	23	46	90	82	118.528
30	24	40	77	60	53.700
30	25	35	66	43	22.331
30	26	32	61	35	8.203
30	27	30	59	30	2.380
30	28	30	60	30	435
30	29	30	61	30	30

Dari Tabel 6.2 dapat dilihat dengan jelas banyaknya jumlah evaluasi kriteria pada keempat algoritma *Branch and Bound*. Algoritma FBB memiliki jumlah evaluasi kriteria yang paling sedikit. Pada algoritma FBB, jumlah evaluasi kriteria

paling banyak terjadi ketika d = 8 dengan jumlah evaluasi kriteria sebesar 7.295. Pada algoritma BBPP, jumlah evaluasi kriteria paling banyak terjadi ketika d = 8 dengan jumlah evaluasi kriteria sebesar 26.102. Pada algoritma IBB, jumlah evaluasi kriteria paling banyak terjadi ketika d = 8 dengan jumlah evaluasi kriteria sebesar 79.899. Pada algoritma BBB, jumlah evaluasi kriteria paling banyak terjadi ketika d = 12 dengan jumlah evaluasi kriteria sebesar 13.039.372.



Gambar 6.1 Hasil evaluasi fungsi kriteria $J(\bar{x}) = \sum_{\xi_i \in x} i^2$ pada algoritma BBB, IBB, BBPP dan FBB

Hasil dari percobaan ini dapat diketahui secara detil pada Lampiran A.

6.4 UJI COBA KEDUA

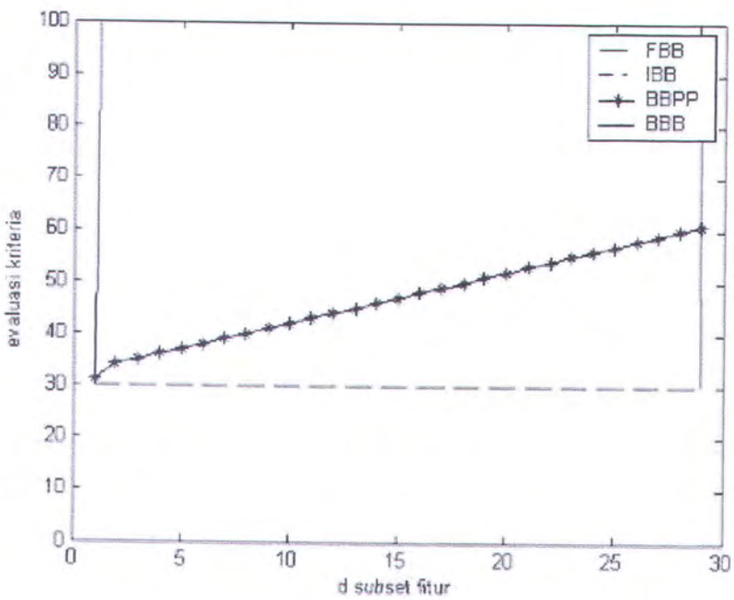
Fungsi kriteria yang digunakan adalah $J(\bar{x}) = \sum_{\xi_i \in x} 2^{(i-1)}$. Hasil dari keempat algoritma *branch and bound* dapat dilihat pada Gambar 6.2 dan Tabel 6.3. Pada

Tabel 6.3, untuk d=13, kolom evaluasi kriteria BBB masih kosong, maksudnya adalah untuk d=13 pada algoritma BBB belum didapatkan jumlah evaluasi kriterianya, namun dengan data yang telah didapat telah menunjukkan bahwa jumlah evaluasi kriteria pada algoritma BBB paling besar.

Tabel 6.3 Hasil Percobaan 2

D	d	Evaluasi kriteria FBB	Evaluasi kriteria BBPP	Evaluasi kriteria IBB	Evaluasi kriteria BBB
30	1	30	31	30	30
30	2	30	34	30	435
30	3	30	35	30	4.060
30	4	30	36	30	24.805
30	5	30	37	30	108.006
30	6	30	38	30	350.389
30	7	30	39	30	870.482
30	8	30	40	30	1.691.130
30	9	30	41	30	2.620.630
30	10	30	42	30	3.312.091
30	11	30	43	30	3.506.700
30	12	30	44	30	3.208.600
30	13	30	45	30	
30	14	30	46	30	1.964.505
30	15	30	47	30	1.382.738
30	16	30	48	30	927.960
30	17	30	49	30	600.174
30	18	30	50	30	376.855
30	19	30	51	30	230.870
30	20	30	52	30	138.414
30	21	30	53	30	81.314
30	22	30	54	30	46.773
30	23	30	55	30	26.250
30	24	30	56	30	14.260
30	25	30	57	30	7.382
30	26	30	58	30	3.531
30	27	30	59	30	1.460
30	28	30	60	30	435
30	29	30	61	30	30

Dari Tabel 6.3 dapat dilihat dengan jelas banyaknya jumlah evaluasi kriteria pada keempat algoritma *Branch and Bound*. Algoritma FBB dan IBB memiliki jumlah evaluasi kriteria yang paling sedikit. Pada algoritma FBB dan IBB, untuk semua nilai d, jumlah evaluasi kriteria sebesar 30. Pada algoritma BBPP, jumlah evaluasi kriteria paling banyak terjadi ketika d = 29 dengan jumlah evaluasi kriteria sebesar 61. Pada algoritma BBB, jumlah evaluasi kriteria paling banyak terjadi ketika d = 11 dengan jumlah evaluasi kriteria sebesar 3.506.700.



Gambar 6.2 Hasil evaluasi fungsi kriteria $J(\bar{x}) = \sum_{\xi_i \in x} 2^{(i-1)}$ pada algoritma BBB, IBB, BBPP dan FBB

Hasil dari percobaan ini dapat diketahui secara detil pada Lampiran B.

6.5 UJI COBA KETIGA

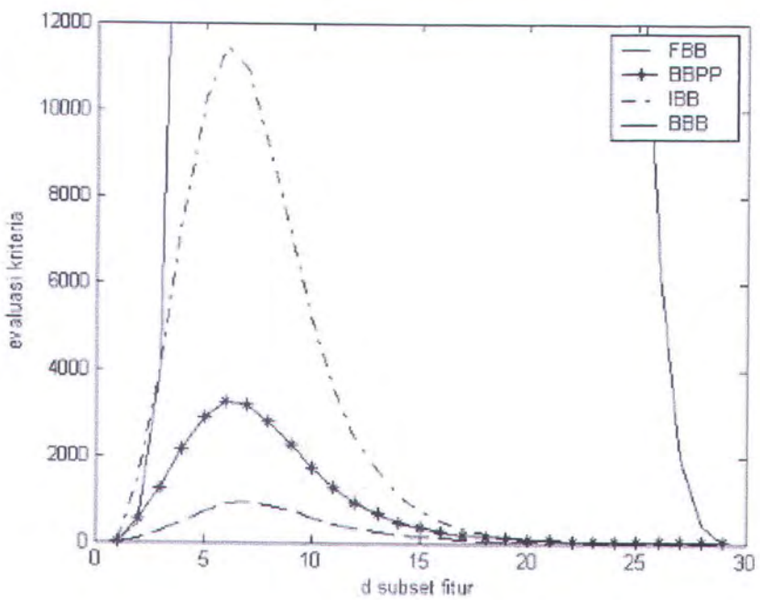
Fungsi kriteria yang digunakan adalah $J(\bar{x}) = \sum_{\xi_i \in x} i^3$. Hasil dari keempat algoritma *branch and bound* dapat dilihat pada Gambar 6.3 dan Tabel 6.4.

Tabel 6.4 Hasil Percobaan 3

D	d	Evaluasi kriteria FBB	Evaluasi kriteria BBPP	Evaluasi kriteria IBB	Evaluasi kriteria BBB
30	1	30	31	30	30
30	2	103	544	1.487	435
30	3	253	1.257	4.042	4.060
30	4	487	2.150	7.452	25.442
30	5	729	2.897	10.276	116.190
30	6	892	3.249	11.456	404.611
30	7	935	3.183	10.927	1.105.804
30	8	859	2.778	9.135	2.421.139
30	9	737	2.275	7.082	4.319.721
30	10	583	1.737	5.091	6.382.471
30	11	444	1.278	3.557	7.938.233
30	12	339	938	2.421	8.458.090
30	13	257	687	1668	7.867.168
30	14	187	484	1104	6.515.240
30	15	143	354	749	4.899.760
30	16	111	262	503	3.406.942
30	17	84	188	330	2.223.984
30	18	67	141	219	1.379.384
30	19	56	113	155	819.782
30	20	46	89	100	469.017
30	21	40	74	72	258.852
30	22	36	67	57	137.594
30	23	35	64	47	70.162
30	24	32	59	37	33.991
30	25	32	60	36	15.367
30	26	30	58	30	6.187
30	27	30	59	30	2.030
30	28	30	60	30	435
30	29	30	61	30	30

Dari Tabel 6.4 dapat dilihat dengan jelas banyaknya jumlah evaluasi kriteria pada keempat algoritma *Branch and Bound*. Algoritma FBB memiliki jumlah

evaluasi kriteria yang paling sedikit. Pada algoritma FBB, jumlah evaluasi kriteria paling banyak terjadi ketika d = 8 dengan jumlah evaluasi kriteria sebesar 859. Pada algoritma BBPP, jumlah evaluasi kriteria paling banyak terjadi ketika d = 6 dengan jumlah evaluasi kriteria sebesar 3.249. Pada algoritma IBB, jumlah evaluasi kriteria paling banyak terjadi ketika d = 6 dengan jumlah evaluasi kriteria sebesar 11.456. Pada algoritma BBB, jumlah evaluasi kriteria paling banyak terjadi ketika d = 12 dengan jumlah evaluasi kriteria sebesar 8.458.090.



Gambar 6.3 Hasil evaluasi fungsi kriteria $J(\bar{x}) = \sum_{\xi_i \in x} i^3$ pada algoritma BBB, IBB, BBPP dan FBB

Hasil dari percobaan ini dapat diketahui secara detil pada Lampiran C.

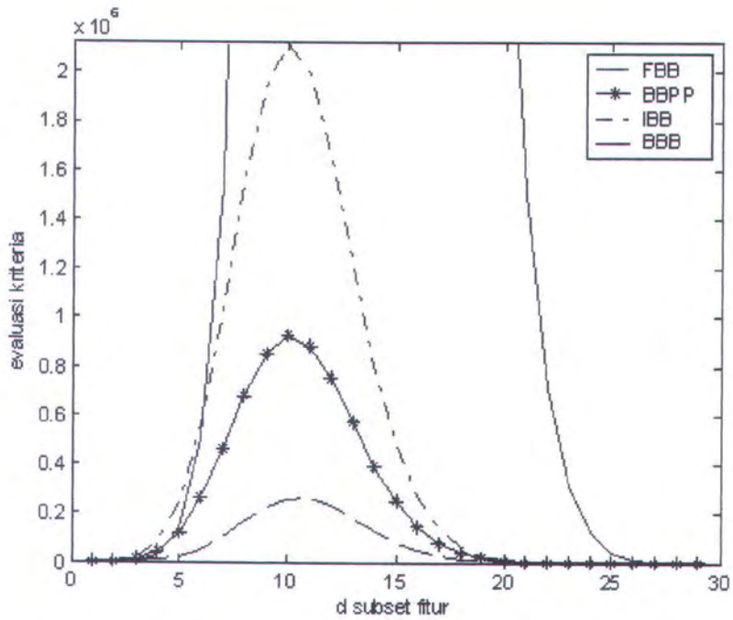
6.6 UJI COBA KEEMPAT

Fungsi kriteria yang digunakan adalah $J(\bar{x}) = \sum_{\xi_i \in x} i$. Hasil dari keempat algoritma *branch and bound* dapat dilihat pada Gambar 6.4 dan Tabel 6.5. Pada Tabel 6.5, untuk d=15 sampai d=18, kolom evaluasi kriteria BBB masih kosong, maksudnya adalah untuk d=13 sampai d=18 pada algoritma BBB belum didapatkan jumlah evaluasi kriterianya, namun dengan data yang telah didapat telah menunjukkan bahwa jumlah evaluasi kriteria pada algoritma BBB paling besar.

Dari Tabel 6.5 dapat dilihat dengan jelas banyaknya jumlah evaluasi kriteria pada keempat algoritma *Branch and Bound.* Algoritma FBB memiliki jumlah evaluasi kriteria yang paling sedikit. Pada algoritma FBB, jumlah evaluasi kriteria paling banyak terjadi ketika d = 11 dengan jumlah evaluasi kriteria sebesar 261.918. Pada algoritma BBPP, jumlah evaluasi kriteria paling banyak terjadi ketika d = 10 dengan jumlah evaluasi kriteria sebesar 930.125. Pada algoritma IBB, jumlah evaluasi kriteria paling banyak terjadi ketika d = 10 dengan jumlah evaluasi kriteria sebesar 2.122.668. Pada algoritma BBB, jumlah evaluasi kriteria paling banyak terjadi ketika d = 13 dengan jumlah evaluasi kriteria sebesar 30.149.059.

Tabel 6.5 Hasil Percobaan 4

D	d	Evaluasi kriteria FBB	Evaluasi kriteria BBPP	Evaluasi kriteria IBB	Evaluasi kriteria BBB
30	1	30	31	30	30
30	2	283	2.691	3.882	435
30	3	1.629	12.913	21.472	4.060
30	4	6.895	45.315	84.148	26.899
30	5	21.698	122.256	246.603	132.354
30	6	52.878	262.810	562.425	505.331
30	7	103.260	462.629	1.029.642	1.544.633
30	8	165.973	681.579	1.551.112	3.861.883
30	9	224.404	855.786	1.962.643	8.027.096
30	10	259.904	930.125	2.122.668	14.056.021
30	11	261.918	887.112	1.992.357	20.976.278
30	12	232.846	751.647	1.644.991	26.961.437
30	13	184.857	572.125	1.209.546	30.149.059
30	14	132.560	395.329	801.137	29.620.752
30	15	86.878	250.634	483.074	
30	16	52.562	147.181	267.847	
30	17	29.552	80.620	137.897	
30	18	15.765	41.906	66.639	
30	19	7.890	20.533	30.403	5.424.158
30	20	3.905	9.886	13364	2.967.370
30	21	1.822	4.504	5.609	1.506.446
30	22	855	2.039	2.301	710.461
30	23	433	976	970	310.339
30	24	193	415	379	124.337
30	25	92	186	152	44.596
30	26	51	98	69	13.556
30	27	35	67	39	3.114
30	28	30	60	30	435
30	29	30	61	30	30



Gambar 6.4 Hasil evaluasi fungsi kriteria $J(\bar{x}) = \sum_{\xi_i \in x} i$ pada algoritma BBB, IBB, BBPP dan FBB

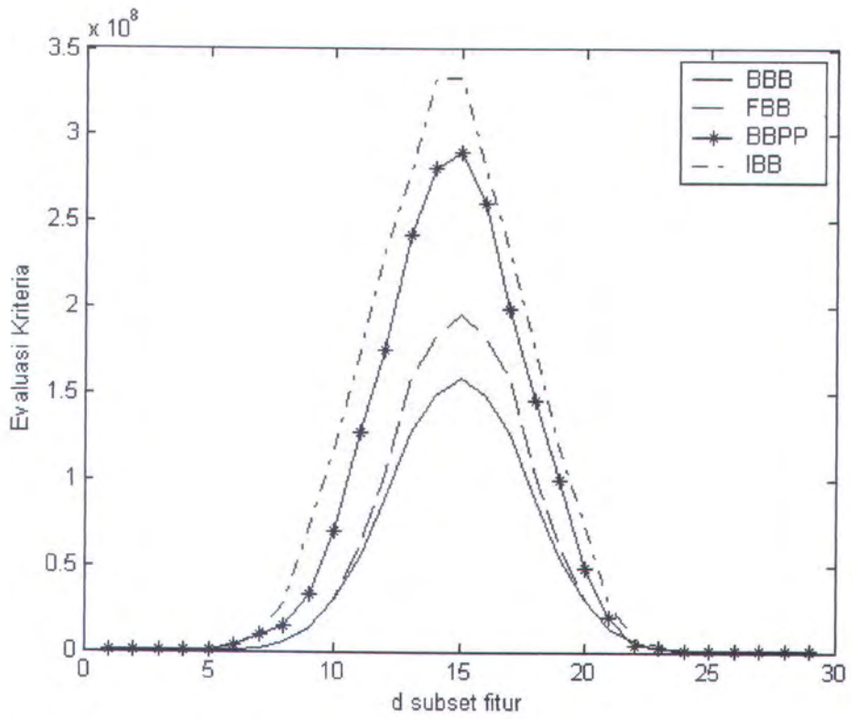
6.7 UJI COBA KELIMA

Fungsi kriteria yang digunakan adalah $J(\bar{x}) = \sum_{\xi_i \in x} |\bar{x}|$. Hasil dari keempat algoritma *branch and bound* dapat dilihat pada Gambar 6.5 dan Tabel 6.6. Dari Tabel 6.6 dapat dilihat dengan jelas banyaknya jumlah evaluasi kriteria pada keempat algoritma *Branch and Bound.* Algoritma BBB memiliki jumlah evaluasi kriteria yang paling sedikit. Hal ini dikarenakan tidak adanya cabang *tree* yang mengalami *cut off.* Pada algoritma BBB, jumlah evaluasi kriteria paling banyak terjadi ketika d = 15 dengan jumlah evaluasi kriteria sebesar 159.375.814. Pada algoritma IBB, jumlah evaluasi kriteria paling banyak terjadi ketika d = 14 dan d=15 dengan jumlah evaluasi kriteria sebesar 332.947.615. Pada algoritma BBPP,

jumlah evaluasi kriteria paling banyak terjadi ketika d = 15 dengan jumlah evaluasi kriteria sebesar 289.782.845. Pada algoritma FBB, jumlah evaluasi kriteria paling banyak terjadi ketika d = 15 dengan jumlah evaluasi kriteria sebesar 195.627.599.

Tabel 6.6 Hasil Percobaan 5

d	Evaluasi kriteria BBB	Evaluasi kriteria IBB	Evaluasi kriteria BBPP	Evaluasi kriteria FBB
1	30	492	466	30
2	435	4.899	4.497	490
3	4.060	35.117	31468	4114
4	27.405	193.660	169.915	27.458
5	142.506	855.034	736.286	142.558
6	593.775	3.104.593	3.223.350	593.832
7	2.035.800	10.295.114	9.924.525	2.035.857
8	5.852.925	28.317.492	15.287.943	5.852.982
9	14.307.150	72.896.225	33.584.110	14.307.207
10	30.045.015	118.417.331	696.71.448	30.251.358
11	54.627.300	174.025.419	127.514.226	61.842.945
12	88.492.583	232.517.426	176.021.415	102.754.623
13	127.501.486	280.258.342	242.517.426	159.375.814
14	148755..429	332.947.615	280.518.226	182.495.347
15	159.375.814	332.947.615	289.782.845	195.627.599
16	148.755.429	280.258.342	260.315.449	182.495.347
17	127.501.486	232.517.426	199.213.516	159.375.814
18	88.492.583	174.025.419	145.375.814	102.754.623
19	54.627.300	118.417.331	99.755.429	61.842.945
20	30.045.015	72.896.225	48.724.155	30.251.358
21	14.307.150	28.317.492	20.323.564	14.307.207
22	5.852.925	7.295.114	3.885.193	5.852.982
23	2.035.800	3.104.593	3.014.682	2.035.857
24	593.775	855.034	736.305	593.832
25	142.506	193.660	169.936	142.538
26	27.405	35.117	31.491	27.436
27	4.060	4.899	4.522	4.090
28	435	492	493	464
29	30	30	60	30



Gambar 6.5 Hasil evaluasi fungsi kriteria $J(\bar{x}) = \sum_{\xi_i \in x} |\bar{x}|$ pada algoritma BBB, IBB, BBPP dan FBB

6.8 EVALUASI UJI COBA

Algoritma *Branch and Bound* sangat dipengaruhi oleh fungsi kriteria yang digunakan dan juga dipengaruhi oleh data yang digunakan. Dari hasil ujucoba yang telah dilakukan menunjukkan bahwa perbedaan fungsi kriteria yang digunakan sangat mempengaruhi *performance* dari algoritma *Branch and Bound.* Berikut merupakan hasil evaluasi terhadap uji coba yang telah dilakukan:

1. Pada percobaan pertama, ketiga dan keempat dapat dilihat bahwa perbedaan pertambahan tiap fitur yang diambil akan menyebabkan adanya *cut off* pada *subtree* yang memiliki nilai kontribusi kriteria

yang lebih kecil, sehingga semua algoritma *Branch and Bound* lebih efektif. Dalam hal ini dapat dilihat pada tabel 6.2, 6.4 dan 6.5 bahwa untuk $d \geq 15$ akan semakin sedikit penghitungan nilai kriteria sebenarnya dilakukan karena pada $d \geq 15$ akan banyak terjadi *cut off*. Pada percobaan ini juga dapat dilihat bahwa prediksi yang digunakan pada algoritma BBPP dan algoritma FBB menyebabkan *performance* kedua algoritma tersebut lebih baik daripada algoritma IBB maupun BBB. Hal ini disebabkan oleh banyaknya prediksi yang dilakukan pada fitur-fitur di tiap *node tree* sehingga tidak perlu dilakukan penghitungan nilai kriteria sebenarnya.

2. Pada percobaan kedua terjadi karena tiap fitur pada cabang pertama menghasilkan kontribusi yang lebih tinggi dari semua fitur pada cabang yang lain, sehingga cabang yang lain ini mengalami *cut off*. *Cut off* yang terjadi sangat cepat sehingga algoritma selesai dalam waktu linier. Hanya pada cabang pertama saja nilai kriteria sebenarnya dihitung.
3. Percobaan kelima merupakan suatu kondisi terburuk yang terjadi pada keempat algoritma *Branch and Bound*. Fungsi kriteria yang digunakan menyebabkan semua fitur menghasilkan nilai kriteria yang sama sehingga tidak ada cabang yang mengalami *cut off*. Sebagai contoh: untuk $d = 1$, semua fitur akan menghasilkan nilai kriteria yang sama yaitu 1, sehingga tidak akan terjadi *cut offf* pada *subtree*, sebab tidak

akan ditemukan satu *node* dengan nilai kriteria lebih kecil dari nilai *bound* yang telah didapat.

4. Nilai kriteria tidak tergantung dari ukuran *subset* pada *level tree*, karena nilai kriteria sangat tergantung pada fungsi kriteria yang digunakan.

BAB VII

SIMPULAN

Dari aplikasi yang telah dibuat beserta uji coba yang telah dilakukan terhadapnya, dapat diambil beberapa kesimpulan sebagai berikut:

1. Algoritma *Branch and Bound* sangat dipengaruhi oleh fungsi kriteria yang digunakan dan juga dipengaruhi oleh data yang digunakan.
2. Prediksi yang digunakan pada algoritma BBPP dan algoritma FBB menyebabkan *performance* kedua algoritma tersebut lebih baik daripada algoritma IBB maupun BBB.
3. Kondisi terburuk yang terjadi pada keempat algoritma *Branch and Bound*, yaitu bila fungsi kriteria yang digunakan menyebabkan semua fitur menghasilkan nilai kriteria yang sama sehingga tidak ada cabang yang mengalami *cut off*.
4. Nilai kriteria sangat tergantung pada fungsi kriteria yang digunakan.

DAFTAR PUSTAKA

[F90] K. Fukunaga, "Introduction to Statistical Pattern Recognition", second ed. Academic Press, Inc., 1990.

[SPK04] P. Somol, P. Pudil dan J. Kittler, "Fast Branch and Bound Algorithm for Optimal Feature Selection", *IEEE Trans. On Pattern Analysis and Machine Intelligence*, vol. 26, No. 7, pp. 900-912, 2004.

[DHS01] R.O. Duda, P.E. Hart dan D.G. Stork, "Pattern Classification", *John Wiley & Sons,Inc*, 2001.

[CF94] R. Caruana dan D. Freitag, "Greedy Attribute Selection", *Proc. Int'l Conf. Machine Learning*, pp. 28-36, 1994.

[JZ97] A.K. Jain dan D. Zongker, "Feature Selection: Evaluation, Application, and Small Sample Performance", *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 19, pp. 153-158, 1997.

[KGV83] S. Kirkpatrick, C.D. Gelatt Jr dan M.P. Vecchi, "Optimization by Simulated Annealing", *Science*, vol. 220, no. 4598, pp. 671-680, 1983.

[LMD98] H. Liu, H. Matoda dan M. Dash, "A Monotonic Measure for Optimal Feature Selection", *Proc. European Conf. Machine Learning*, pp. 101-106, 1998.

[NF77] P.M. Narendra dan K. Fukunaga, "A Branch and Bound Algorithm for Feature Subset Selection", *IEEE Trans. Computers*, vol. 26, no. 9, pp. 917-922, Sept. 1977.

[SPFK00] P. Somol, P. Pudil, F.J. Ferri dan J. Kittler, "Fast Branch and Bound Algorithm in Feature Selection", *Proc. Fourth World Multiconf. Systemics, Cybernetics, and Informatics*, vol. 7, Part 1, pp. 646-651, 2000.

[YY93] B. Yu dan B. Yuan, "A More Efficient Branch and Bound Algorithm for Feature Selection", *Pattern Recognition*, vol. 26, pp. 883-889, 1993.

[NIST] http://www.nist.gov

LAMPIRAN A

HASIL PERCOBAAN 1

Cara Baca Tabel Lampiran:

1. kolom "d" merupakan kolom jumlah fitur yang diambil.
2. kolom "evaluasi" merupakan kolom untuk menyimpan jumlah evaluasi fungsi kriteria yang terjadi pada algoritma *Branch and Bound*.
3. kolom "bound" merupakan kolom untuk menyimpan nilai bound.
4. kolom "x" merupakan kolom untuk menyimpan fitur-fitur yang diambil.
5. kolom "waktu iterasi perfitur" merupakan kolom untuk menyimpan lamanya waktu yang dibutuhkan untuk mendapatkan hasilnya tiap fitur.

Contoh:

Pada tabel A.1, misalkan untuk d = 1 maka jumlah evaluasi fungsi kriteria adalah 30, nilai bound yang didapat sebesar 900, fitur yang diambil adalah fitur 30 dan waktu iterasinya sebesar 0.32800 detik.

Tabel A.1 Hasil percobaan fungsi kriteria $J(\bar{x}) = \sum_{\xi_i \in x} i^2$ terhadap algoritma BBB

d	evaluasi	bound	x	waktu iterasi per fitur (detik)
1	30	900	30	0.32800
2	435	1741	29 30	4.59400
3	4060	2525	28 29 30	24.70300
4	25946	3254	27 28 29 30	121.27000
5	121822	3930	26 27 28 29 30	495.25000
6	439336	4555	25 26 27 28 29 30	1618.20000
7	1252242	5131	24 25 26 27 28 29 30	3699.40000
8	2881194	5660	23 24 25 26 27 28 29 30	6440.90000
9	5443041	6144	22 23 24 25 26 27 28 29 30	10797.00000
10	8571300	6585	21 22 23 24 25 26 27 28 29 30	15320.00000
11	11413455	6985	20 21 22 23 24 25 26 27 28 29 30	18695.00000
12	13039372	7346	19 20 21 22 23 24 25 26 27 28 29 30	19865.00000
13	12974378	7670	18 19 20 21 22 23 24 25 26 27 28 29 30	20937.00000
14	11420335	7959	17 18 19 20 21 22 23 24 25 26 27 28 29 30	15536.00000
15	9034241	8215	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	11836.00000
16	6522050	8440	15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	8260.10000
17	4355601	8636	14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	5387.60000
18	2723591	8805	13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	3292.50000
19	1607928	8949	12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	1910.40000
20	902198	9070	11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	1056.60000

d	evaluasi	bound	x	waktu iterasi per fitur (detik)
21	482220	9170	10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	561.31000
22	245592	9251	9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	281.63000
23	118528	9315	8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	132.58000
24	53700	9364	7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	60.39100
25	22331	9400	6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	24.32800
26	8203	9425	5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	9.10900
27	2380	9441	4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	2.65700
28	435	9450	3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	0.43700
29	30	9454	2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	0.03100

Total waktu yang dibutuhkan sebesar 146370,31800 detik.

Tabel A.2 Hasil percobaan fungsi kriteria $J(\bar{x}) = \sum_{\xi_i \in x} i^2$ terhadap algoritma IBB

d	evaluasi	bound	x	waktu iterasi per fitur (detik)
1	30	900	30	0.36000
2	2441	1741	29 30	9.17200
3	9157	2525	28 29 30	27.51500
4	23394	3254	27 28 29 30	83.88900
5	44192	3930	26 27 28 29 30	159.26000
6	65211	4555	25 26 27 28 29 30	273.78000
7	78472	5131	24 25 26 27 28 29 30	401.23000
8	79899	5660	23 24 25 26 27 28 29 30	515.65000
9	70887	6144	22 23 24 25 26 27 28 29 30	628.64000

d	evaluasi	bound	x	waktu iterasi per fitur (detik)
10	56167	6585	21 22 23 24 25 26 27 28 29 30	73.392.000
11	40600	6985	20 21 22 23 24 25 26 27 28 29 30	79.563.000
12	27429	7346	19 20 21 22 23 24 25 26 27 28 29 30	83.941.000
13	17474	7670	18 19 20 21 22 23 24 25 26 27 28 29 30	86.998.000
14	10700	7959	17 18 19 20 21 22 23 24 25 26 27 28 29 30	88.230.000
15	6330	8215	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	89.192.000
16	3705	8440	15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	89.758.000
17	2134	8636	14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	90.292.000
18	1200	8805	13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	90.495.000
19	701	8949	12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	90.606.000
20	401	9070	11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	90.687.000
21	223	9170	10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	90.719.000
22	148	9251	9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	90.748.000
23	82	9315	8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	90.759.000
24	60	9364	7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	90.770.000
25	43	9400	6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	90.776.000
26	35	9425	5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	90.784.000
27	30	9441	4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	90.787.000
28	30	9450	3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	90.790.000
29	30	9454	2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	90.794.000

Total waktu yang dibutuhkan sebesar 19800,30600 detik.

Tabel A.3 Hasil percobaan fungsi kriteria $J(\bar{x}) = \sum_{\xi_i \in x} i^2$ terhadap algoritma BBPP

d	evaluasi	bound	x	waktu iterasi per fitur (detik)
1	31	900	30	0.40600
2	1146	1741	29 30	12.49900
3	3618	2525	28 29 30	37.79600
4	8310	3254	27 28 29 30	95.42000
5	14743	3930	26 27 28 29 30	200.21000
6	21120	4555	25 26 27 28 29 30	331.87000
7	25289	5131	24 25 26 27 28 29 30	483.12000
8	26102	5660	23 24 25 26 27 28 29 30	612.71000
9	23792	6144	22 23 24 25 26 27 28 29 30	741.13000
10	19586	6585	21 22 23 24 25 26 27 28 29 30	834.20000
11	14851	6985	20 21 22 23 24 25 26 27 28 29 30	898.83000
12	10592	7346	19 20 21 22 23 24 25 26 27 28 29 30	949.53000
13	7161	7670	18 19 20 21 22 23 24 25 26 27 28 29 30	972.93000
14	4670	7959	17 18 19 20 21 22 23 24 25 26 27 28 29 30	992.09000
15	2945	8215	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	1001.10000
16	1867	8440	15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	1004.70000
17	1173	8636	14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	1006.80000
18	706	8805	13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	1008.40000
19	449	8949	12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	1009.40000
20	283	9070	11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	1010.00000
21	177	9170	10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	1010.40000
22	133	9251	9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	1011.20000
23	90	9315	8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	1011.30000
24	77	9364	7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	1011.40000

d	evaluasi	bound	x	waktu iterasi per fitur (detik)
25	66	9400	6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	1011.50000
26	61	9425	5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	1011.50000
27	59	9441	4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	1011.50000
28	60	9450	3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	1011.60000
29	61	9454	2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	1011.60000

Total waktu yang dibutuhkan sebesar 22305,14100 detik.

Tabel A.4 Hasil percobaan fungsi kriteria $J(\bar{x}) = \sum_{\xi_i \in x} i^2$ terhadap algoritma FBB

d	evaluasi	bound	x	waktu iterasi per fitur (detik)
1	30	900	30	0.39100
2	161	1741	29 30	3.21900
3	576	2525	28 29 30	10.93700
4	1556	3254	27 28 29 30	31.56200
5	3162	3930	26 27 28 29 30	64.95200
6	5050	4555	25 26 27 28 29 30	105.15000
7	6592	5131	24 25 26 27 28 29 30	170.79000
8	7295	5660	23 24 25 26 27 28 29 30	238.75000
9	7035	6144	22 23 24 25 26 27 28 29 30	298.24000
10	6062	6585	21 22 23 24 25 26 27 28 29 30	345.68000
11	4777	6985	20 21 22 23 24 25 26 27 28 29 30	379.12000
12	3524	7346	19 20 21 22 23 24 25 26 27 28 29 30	402.46000
13	2450	7670	18 19 20 21 22 23 24 25 26 27 28 29 30	417.16000
14	1638	7959	17 18 19 20 21 22 23 24 25 26 27 28 29 30	426.51000
15	1057	8215	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	432.07000
16	692	8440	15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	435.62000
17	448	8636	14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	437.71000
18	278	8805	13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	438.91000
19	184	8949	12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	439.58000
20	122	9070	11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	440,08000
21	81	9170	10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	440,38000
22	65	9251	9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	440,65000
23	46	9315	8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	440,82000
24	40	9364	7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	440,90000
25	35	9400	6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	440,99000
26	32	9425	5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	441,05000

d	evaluasi	bound	x	waktu iterasi per fitur (detik)
27	30	9441	4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	441,12000
28	30	9450	3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	441,18000
29	30	9454	2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	441,21000

Total waktu yang dibutuhkan sebesar 9487,19100 detik.

LAMPIRAN B

HASIL PERCOBAAN 2

Tabel B.1 Hasil percobaan fungsi kriteria $J(\bar{x}) = \sum_{\xi_i \in x} 2^{(i-1)}$ terhadap algoritma BBB

d	evaluasi	bound	x	waktu iterasi per fitur (detik)
1	30	536870000	30	0.5940
2	435	805310000	29 30	3.4530
3	4060	939520000	28 29 30	20.2500
4	24805	1006600000	27 28 29 30	90.2630
5	108006	1040200000	26 27 28 29 30	306.0730
6	350389	1057000000	25 26 27 28 29 30	796.3440
7	870482	1065400000	24 25 26 27 28 29 30	1665.8000
8	1691130	1069500000	23 24 25 26 27 28 29 30	2752.9000
9	2620630	1071600000	22 23 24 25 26 27 28 29 30	
10	3312091	1072700000	21 22 23 24 25 26 27 28 29 30	
11	3506700	1073200000	20 21 22 23 24 25 26 27 28 29 30	
12	3.208600	1073500000	19 20 21 22 23 24 25 26 27 28 29 30	
13		1073600000	18 19 20 21 22 23 24 25 26 27 28 29 30	
14	1964505	1073700000	17 18 19 20 21 22 23 24 25 26 27 28 29 30	
15	1382738	1073700000	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	1295.4000
16	927960	1073700000	15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	885.5770
17	600174	1073700000	14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	539.7560
18	376855	1073700000	13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	326.3380
19	230870	1073700000	12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	198.9030
20	138414	1073700000	11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	118.1380

d	evaluasi	bound	x	waktu iterasi per fitur (detik)
21	81314	1073700000	10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	69.6860
22	46773	1073700000	9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	40.4060
23	26250	1073700000	8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	23.0150
24	14260	1073700000	7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	12.8750
25	7382	1073700000	6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	6.9690
26	3531	1073700000	5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	3.6570
27	1460	1073700000	4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	1.5150
28	435	1073700000	3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	0.5940
29	30	1073700000	2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	0.0630

Total waktu yang dibutuhkan sebesar 9158,5690 detik.

Tabel B.2 Hasil percobaan fungsi kriteria $J(\bar{x}) = \sum_{\xi_i \in x} 2^{(i-1)}$ terhadap algoritma IBB

d	evaluasi	bound	x	waktu iterasi per fitur (detik)
1	30	536870000	30	0.35900
2	30	805310000	29 30	0.73400
3	30	939520000	28 29 30	1.04700
4	30	1006600000	27 28 29 30	1.31200
5	30	1040200000	26 27 28 29 30	1.45300
6	30	1057000000	25 26 27 28 29 30	1.59400
7	30	1065400000	24 25 26 27 28 29 30	1.71900
8	30	1069500000	23 24 25 26 27 28 29 30	1.84400
9	30	1071600000	22 23 24 25 26 27 28 29 30	1.95300
10	30	1072700000	21 22 23 24 25 26 27 28 29 30	2.06200
11	30	1073200000	20 21 22 23 24 25 26 27 28 29 30	2.15600
12	30	1073500000	19 20 21 22 23 24 25 26 27 28 29 30	2.25000
13	30	1073600000	18 19 20 21 22 23 24 25 26 27 28 29 30	2.32800
14	30	1073700000	17 18 19 20 21 22 23 24 25 26 27 28 29 30	2.40600
15	30	1073700000	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	2.48400
16	30	1073700000	15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	2.53100
17	30	1073700000	14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	2.59400
18	30	1073700000	13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	2.64000
19	30	1073700000	12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	2.68700
20	30	1073700000	11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	2.73400

d	evaluasi	bound	x	waktu iterasi per fitur (detik)
21	223	9170	10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	907.19000
22	148	9251	9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	907.48000
23	82	9315	8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	907.59000
24	60	9364	7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	907.70000
25	43	9400	6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	907.76000
26	35	9425	5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	907.84000
27	30	9441	4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	907.87000
28	30	9450	3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	907.90000
29	30	9454	2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	907.94000

Total waktu yang dibutuhkan sebesar 64,65100 detik.

Tabel B.3 Hasil percobaan fungsi kriteria $J(\bar{x}) = \sum_{\xi_i \in x} 2^{(i-1)}$ terhadap algoritma BBPP

d	evaluasi	bound	x	waktu iterasi per fitur (detik)
1	31	536870000	30	0.43700
2	34	805310000	29 30	0.84400
3	35	939520000	28 29 30	1.04700
4	36	1006600000	27 28 29 30	1.23400
5	37	1040200000	26 27 28 29 30	1.42200
6	38	1057000000	25 26 27 28 29 30	1.59400
7	39	1065400000	24 25 26 27 28 29 30	1.75000
8	40	1069500000	23 24 25 26 27 28 29 30	1.89000
9	41	1071600000	22 23 24 25 26 27 28 29 30	2.03100
10	42	1072700000	21 22 23 24 25 26 27 28 29 30	2.15600
11	43	1073200000	20 21 22 23 24 25 26 27 28 29 30	2.28100
12	44	1073500000	19 20 21 22 23 24 25 26 27 28 29 30	2.39000
13	45	1073600000	18 19 20 21 22 23 24 25 26 27 28 29 30	2.48400
14	46	1073700000	17 18 19 20 21 22 23 24 25 26 27 28 29 30	2.57800
15	47	1073700000	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	2.65600
16	48	1073700000	15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	2.73400
17	49	1073700000	14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	2.81200
18	50	1073700000	13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	3.26500
19	51	1073700000	12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	3.31200
20	52	1073700000	11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	3.35900
21	53	1073700000	10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	3.40600
22	54	1073700000	9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	3.45300
23	55	1073700000	8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	3.48400
24	56	1073700000	7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	3.51500

d	evaluasi	bound	x																																				waktu iterasi per fitur (detik)
25	57	1073700000	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	0	0	0	0	3.54700
26	58	1073700000	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	0	0	0	3.56200
27	59	1073700000	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	0	0	3.59300
28	60	1073700000	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	0	3.60900
29	61	1073700000	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	3.62500

Total waktu yang dibutuhkan sebesar 56,13400 detik.

Tabel B.4 Hasil percobaan fungsi kriteria $J(\bar{x}) = \sum_{\xi_i \in x} 2^{(i-1)}$ terhadap algoritma FBB

d	evaluasi	bound	x	waktu iterasi per fitur
1	30	900	30	0.39100
2	161	1741	29 30	3.21900
3	576	2525	28 29 30	10.93700
4	1556	3254	27 28 29 30	31.56200
5	3162	3930	26 27 28 29 30	64.95200
6	5050	4555	25 26 27 28 29 30	105.15000
7	6592	5131	24 25 26 27 28 29 30	170.79000
8	7295	5660	23 24 25 26 27 28 29 30	238.75000
9	7035	6144	22 23 24 25 26 27 28 29 30	298.24000
10	6062	6585	21 22 23 24 25 26 27 28 29 30	345.68000
11	4777	6985	20 21 22 23 24 25 26 27 28 29 30	379.12000
12	3524	7346	19 20 21 22 23 24 25 26 27 28 29 30	402.46000
13	2450	7670	18 19 20 21 22 23 24 25 26 27 28 29 30	417.16000
14	1638	7959	17 18 19 20 21 22 23 24 25 26 27 28 29 30	426.51000
15	1057	8215	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	432.07000
16	692	8440	15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	435.62000
17	448	8636	14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	437.71000
18	278	8805	13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	438.91000
19	184	8949	12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	439.58000
20	122	9070	11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	440.08000

d	evaluasi	bound	x																												waktu iterasi per fitur
21	81	9170	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	0	0	0	0	0	0	0	0	440.38000
22	65	9251	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	0	0	0	0	0	0	0	440.65000
23	46	9315	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	0	0	0	0	0	0	440.82000
24	40	9364	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	0	0	0	0	0	440.90000
25	35	9400	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	0	0	0	0	440.99000
26	32	9425	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	0	0	0	441.05000
27	30	9441	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	0	0	441.12000
28	30	9450	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	0	441.18000
29	30	9454	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	441.21000

LAMPIRAN C

HASIL PERCOBAAN 3

Tabel C.1 Hasil percobaan fungsi kriteria $J(\bar{x}) = \sum_{\xi_i \in x} i^3$ terhadap algoritma BBB

d	evaluasi	bound	x	waktu iterasi per fitur (detik)
1	30	27000	30	0.5470
2	435	51389	29 30	4.0000
3	4060	73341	28 29 30	29.4060
4	25442	93024	27 28 29 30	136.9190
5	116190	110600	26 27 28 29 30	
6	404611	126225	25 26 27 28 29 30	
7	1105804	140049	24 25 26 27 28 29 30	
8	2421139	152216	23 24 25 26 27 28 29 30	
9	4319721	162864	22 23 24 25 26 27 28 29 30	
10	6382471	172125	21 22 23 24 25 26 27 28 29 30	
11	7938233	180125	20 21 22 23 24 25 26 27 28 29 30	
12	8458090	186984	19 20 21 22 23 24 25 26 27 28 29 30	
13	7867168	192816	18 19 20 21 22 23 24 25 26 27 28 29 30	
14	6515240	197729	17 18 19 20 21 22 23 24 25 26 27 28 29 30	
15	4899760	201825	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	
16	3406942	205200	15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	
17	2223984	207944	14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	
18	1379384	210141	13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	
19	819782	211869	12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	
20	469017	213200	11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	

d	evaluasi	bound	x	waktu iterasi per fitur (detik)
21	258852	214200	10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	
22	137594	214929	9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	
23	70162	215441	8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	
24	33991	215784	7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	
25	15367	216000	6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	
26	6187	216125	5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	22.4990
27	2030	216189	4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	11.5470
28	435	216216	3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	3.8750
29	30	216224	2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	0.8120
				0.0940

Total waktu yang dibutuhkan sebesar 209,6990 detik.

Tabel C.2 Hasil percobaan fungsi kriteria $J(\bar{x}) = \sum_{\xi_i \in x} i^3$ terhadap algoritma IBB

d	evaluasi	bound	x	waktu iterasi per fitur (detik)
1	30	27000	30	0.37500
2	1487	51389	29 30	5.85900
3	4042	73341	28 29 30	15.50000
4	7452	93024	27 28 29 30	31.89000
5	10276	110600	26 27 28 29 30	48.93700
6	11456	126225	25 26 27 28 29 30	64.84300
7	10927	140049	24 25 26 27 28 29 30	79.12400
8	9135	152216	23 24 25 26 27 28 29 30	87.81100
9	7082	162864	22 23 24 25 26 27 28 29 30	95.87300
10	5091	172125	21 22 23 24 25 26 27 28 29 30	100.80000
11	3557	180125	20 21 22 23 24 25 26 27 28 29 30	104.17000
12	2421	186984	19 20 21 22 23 24 25 26 27 28 29 30	106.34000
13	1668	192816	18 19 20 21 22 23 24 25 26 27 28 29 30	107.83000
14	1104	197729	17 18 19 20 21 22 23 24 25 26 27 28 29 30	109.26000
15	749	201825	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	110.25000
16	503	205200	15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	111.06000
17	330	207944	14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	111.53000
18	219	210141	13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	111.86000
19	155	211869	12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	112.05000
20	100	213200	11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	112.22000

d	evaluasi	bound	x	waktu iterasi per fitur (detik)
21	72	214200	10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	112.31000
22	57	214929	9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	112.37000
23	47	215441	8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	112.44000
24	37	215784	7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	112.48000
25	36	216000	6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	112.53000
26	30	216125	5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	112.56000
27	30	216189	4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	112.59000
28	30	216216	3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	112.61000
29	30	216224	2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	112.61000

Total waktu yang dibutuhkan sebesar 2640,08200 detik.

Tabel C.3 Hasil percobaan fungsi kriteria $J(\bar{x}) = \sum_{\xi_i \in x} i^3$ terhadap algoritma BBPP

d	evaluasi	bound	x	waktu iterasi per fitur (detik)
1	31	27000	30	0.43800
2	544	51389	29 30	5.17200
3	1257	73341	28 29 30	13.53100
4	2150	93024	27 28 29 30	26.82800
5	2897	110600	26 27 28 29 30	46.37400
6	3249	126225	25 26 27 28 29 30	66.04600
7	3183	140049	24 25 26 27 28 29 30	88.76400
8	2778	152216	23 24 25 26 27 28 29 30	106.08000
9	2275	162864	22 23 24 25 26 27 28 29 30	119.79000
10	1737	172125	21 22 23 24 25 26 27 28 29 30	129.19000
11	1278	180125	20 21 22 23 24 25 26 27 28 29 30	136.54000
12	938	186984	19 20 21 22 23 24 25 26 27 28 29 30	140.59000
13	687	192816	18 19 20 21 22 23 24 25 26 27 28 29 30	143.47000
14	484	197729	17 18 19 20 21 22 23 24 25 26 27 28 29 30	145.26000
15	354	201825	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	146.56000
16	262	205200	15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	147.48000
17	188	207944	14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	148.07000
18	141	210141	13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	148.50000
19	113	211869	12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	148.76000
20	89	213200	11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	148.95000
21	74	214200	10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	149.09000
22	67	214929	9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	149.15000
23	64	215441	8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	149.25000
24	59	215784	7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	149.29000

d	evaluasi	bound	x	waktu iterasi per fitur (detik)
25	60	216000	6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	149.34000
26	58	216125	5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	149.37000
27	59	216189	4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	149.40000
28	60	216216	3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	149.42000
29	61	216224	2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	149.47000

Total waktu yang dibutuhkan sebesar 2653,17300 detik.

Tabel C.4 Hasil percobaan fungsi kriteria $J(\bar{x}) = \sum_{\xi_i \in x} i^3$ terhadap algoritma FBB

d	evaluasi	bound	x	waktu iterasi per fitur (detik)
1	30	27000	30	0.39000
2	103	51389	29 30	2.64000
3	253	73341	28 29 30	7.15600
4	487	93024	27 28 29 30	14.21800
5	729	110600	26 27 28 29 30	19.99900
6	892	126225	25 26 27 28 29 30	26.65600
7	935	140049	24 25 26 27 28 29 30	33.51500
8	859	152216	23 24 25 26 27 28 29 30	39.71800
9	737	162864	22 23 24 25 26 27 28 29 30	43.43600
10	583	172125	21 22 23 24 25 26 27 28 29 30	46.03000
11	444	180125	20 21 22 23 24 25 26 27 28 29 30	47.78000
12	339	186984	19 20 21 22 23 24 25 26 27 28 29 30	48.96800
13	257	192816	18 19 20 21 22 23 24 25 26 27 28 29 30	49.79600
14	187	197729	17 18 19 20 21 22 23 24 25 26 27 28 29 30	50.46800
15	143	201825	16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	50.87400
16	111	205200	15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	51.17100
17	84	207944	14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	51.38900
18	67	210141	13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	51.54600
19	56	211869	12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	51.67100
20	46	213200	11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	51.74900

d	evaluasi	bound	x	waktu iterasi per fitur
21	40	214200	10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	51.81100
22	36	214929	9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	51.87400
23	35	215441	8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	51.92100
24	32	215784	7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	51.95200
25	32	216000	6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	51.98300
26	30	216125	5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	52.01400
27	30	216189	4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	52.03000
28	30	216216	3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	52.06100
29	30	216224	2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30	52.07700

Total waktu yang dibutuhkan sebesar 1206,89300 detik.