Pendimensian Node Hardware Pada Jaringan SDH (Synchronous Digital Hierarchy) dengan Metode MILP, Heuristic dan Variable Fixation Test

M. Zen Samsono Hadi, Aries Pratiarso, dan M. Agus Zainuddin

Abstrak—Optimasi biaya merupakan hal yang paling krusial dalam perencanaan jaringan telekomunikasi, yang akan menentukan berapa banyak *port card*, *pluggable device kind* dan *base equipment configuration* (BEC) yang akan diinstal pada sebuah *node hardware* serta biaya minimal yang diperlukan. Di dalam paper ini akan digunakan 2 metode untuk menyelesaikan permasalahan optimasi tersebut yaitu *Mixed Integer Linear Programming* (MILP) dan *heuristic method*.Pengembangan algoritma dalam hal ini juga digunakan untuk melakukan *preprocessing* dengan *variable fixation test* yang akan menghapus beberapa variabe yaitu nilai BEC, yang sebenarnya bisa dihilangkan untuk mempercepat proses perhitungan. Untuk itu akan dilakukan proses penggabungan 2 metode diatas, dimana *heuristic method* digunakan untuk menghapus beberapa nilai BEC tersebut dan hasilnya akan dimasukkan ke algoritma MILP agar mendapatkan hasil yang optimal. Hasil yang didapat bahwa *heuristic method* sangat efektif untuk mendelete beberapa variabel dalam BEC, sehingga mempercepat proses yang dilakukan oleh metode MILP. Waktu komputasi metode heuristic cenderung stabil pada 0.01s, sedangkan MILP cenderung eksponensial ketika datanya semakin kompleks.

Kata Kunci—Node Hardware, MILP, metode Heuristic, fixation test.

1 PENDAHULUAN

SAAT ini pengembangan jaringan telekomunikasi tersebar sangat luas. Dan tentunya pengembangan jaringan tersebut harus didukung oleh *node hardware* yang efisien.

Node hardware yang efisien membutuhkan biaya yang minimal dalam membangun sebuah *node hardware*. Sebuah *node hardware* sendiri terdiri dari 3 bagian penting yaitu *port card*, *pluggable device* dan *base equipment configuration* yang akan menangani aliran bit trafik pada node tersebut.

Untuk menyelesaikan permasalahan diatas digunakan metode MILP dan *heuristic*. Metode MILP sangat cepat dan dapat memberikan solusi yang optimal jika data masukannya adalah kecil dan tidak banyak. Tetapi jika datanya besar dan kompleks, metode tersebut akan membutuhkan waktu yang lama dikarenakan banyaknya *branch-and-bound* [1][2]. Permasalahan tersebut dapat diatasi dengan penggabungan metode MILP dan *heuristic* [3]-[5]. *Heuristic* digunakan untuk mendapatkan solusi yang mendekati optimal dan cepat, dan hasilnya dipakai untuk menghapus beberapa variabel dari data masukan sebelum diselesaikan dengan MILP.

Metode MILP sudah teruji untuk mengatasi permasalahan dimensi *node hardware*[6], dan penggunaan metode *heuristic* dan perbandingan kecepatannya antara 2 metode diatas sudah dikembangkan [7], sehingga dalam makalah ini akan dilakukan penggabungan 2 metode diatas.

- *M.Zen Samsono Hadi, Jurusan Teknik Telekomunikasi, Politeknik Elektronika Negeri Surabaya , Jl. Raya ITS Keputih Sukolilo, Telp:031-5947280. E-mail: zenhadi@eepis-its.edu.*
- *Aries Pratiarso dan M. Agus Zainuddin, Jurusan Teknik Telekomunikasi, Politeknik Elektronika Negeri Surabaya , Jl. Raya ITS Keputih Sukolilo, Telp:031-5947280/ext:4109, fax:031-5946114. E-mail:aries@eepis-its.edu, magusz@eepis-its.edu.*

2 METODOLOGI

2.1 Jaringan SDH

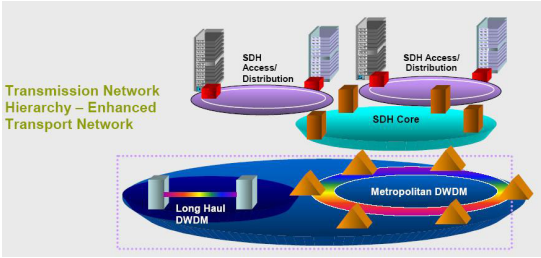
SDH (*Synchronous Digital Hierarchy*) adalah standar internasional untuk telekomunikasi berkecepatan tinggi melalui jaringan optik atau listrik yang dapat mengirimkan sinyal digital dengan kapasitas yang bervariasi. SDH adalah sistem sinkron yang menyediakan infrastruktur jaringan yang lebih fleksibel. Jaringan SDH dapat langsung menambah sinyal frekuensi rendah menjadi sinyal frekuensi tinggi atau mendrop kecepatan yang terlalu tinggi tanpa harus melakukan *multiplexing/demultiplexing*.

Standar SDH pertama di setujui oleh badan ITU-T pada bulan nopember 1988 dan di rekomendasikan oleh G707,G708 serta G709 dan di publikasikan di CCITT *Blue book* pada tahun 1989. Badan ini menentukan *rate,frame*,dan proses *multiplexing*. Kemudian SDH di jadikan sebagai standar jaringan telekomunikasi kecepatan tinggi internasional.

Jaringan sinkron SDH memiliki perbedaan dengan jaringan PDH (*Plesiochronous Digital Hierarchy*) yaitu pada kecepatan pengiriman data yang disinkronkan dengan ketat pada seluruh jaringan, dimungkinkan dengan penggunaan jam atom. Sistem sinkron ini memungkinkan seluruh jaringan antar negara untuk beroperasi secara sinkron atau seiring, dengan menekan jumlah *buffer* yang dibutuhkan antar elemen jaringan [8][9].

SDH memiliki beberapa kelebihan [9]:

1) Standar format digital pertama di dunia.
2) Antarmuka optik pertama.
3) Kompatibilitas *transversal* dapat menekan biaya jaringan. Persainga antar *vendor* akan mengendalikan harga.
4) Struktur multipleks yang sinkron dan fleksibel.
5) *Add-and-drop* lalu lintas data dan kapabilitas *cross connect* yang mudah dan efisien.
6) Pengurangan jumlah *interface back-to-back* dapat meningkatkan ketahanan dan layanan jaringan.
7) Kapabilitas manajemen jaringan yang handal.
8) Arsitektur jaringan baru yang lebih fleksibel dan kemampuan perbaikan jaringan secara otomatis.



Gambar 1. Jaringan telekomunikasi berbasis SDH[9]

2.2 Metode MILP Dan *Heuristic*

MILP menggunakan metode *simplex* dan *branch-and-bound* dalam memecahkan permasalahan *integer programming*. Sedangkan metode *heuristic* adalah teknik pencarian yang akan mencari nilai yang mendekati nilai optimal dengan waktu perhitungan yang cepat tetapi tidak menjamin solusi yang optimal.

2.2.1 Metode Simplex

Metode *simplex* merupakan suatu metode untuk dapat menyelesaikan persamaan LP yang mempunyai variabel keputusan lebih dari dua. Ketentuan dalam menggunakan metode *simplex*[12]:

1) Nilai kanan (NK/RHS) fungsi tujuan harus nol (0).
2) Nilai kanan (RHS) fungsi kendala harus positif. Apabila negative nilai itu harus dikalikan minus satu (-1).
3) Fungsi kendala dengan tanda "$\leq$" harus di ubah ke bentuk "$=$" dengan menambah variabel slack/surplus.
4) Fungsi kendala dengan tanda "$\geq$" diubah ke dalam bentuk "$\leq$" dengan cara mengalikan dengan -1,lalu diubah ke bentuk persamaan dengan di tambahkan variabel slack. Kemudian karena RHSnya negatif,dikalikan lagi dengan -1 dan ditambah artificial variabel (A).
5) Fungsi kendala dengan tanda "=" harus ditambah artificial variabel (A).

2.2.2 Metode Branch-and-Bound

Metode *Branch-and-Bound* telah menjadi kode komputer standar untuk *integer programming*, dan penerapannya dalam praktek metode ini

sangat efisien.Teknik ini dapat diterapkan dengan baik untuk masalah *mixed integer programming*.

Langkah-langkah metode *Branch-and-Bound* untuk masalah optimasi dapat dilakukan seperti berikut[10]:

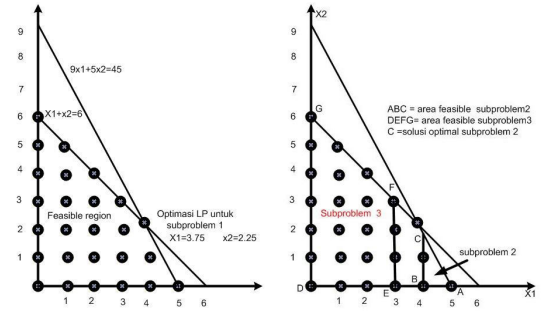
1) Selesaikan *Linier Programming* dengan metode *simpleks*.
2) Amati hasil solusi optimasinya. Jika variabel yang diharapkan adalah bilangan bulat, solusi optimasi sudah tercapai.
3) Nilai solusi kemudian dipecah ke dalam sub-sub masalah. Tujuannya adalah untuk menghilangkan solusi yang tidak memenuhi persyaratan bilangan bulat.
4) Untuk setiap sub-masalah, nilai solusi optimasi kontinyu fungsinya ditetapkan sebagai batas atas. Solusi bilangan bulat terbaik menjadi batas bawah (pada awalnya, ini adalah solusi kontinyu yang dibulatkan ke bawah). Sub-sub masalah yang batas atasnya kurang dari batas bawah tidak diikut sertakan pada analisa selanjutnya. Suatu solusi bilangan bulat adalah batas atas untuk setiap sub masalah yang dicari. Jika solusi yang terjadi demikian, suatu sub masalah dengan batas atas terbaik dipilih untuk dicabangkan. Kembali ke langkah 3.

Untuk memperoleh gambaran yang lebih jelas tentang metode *Branch-and-Bound*, perhatikan contoh masalah berikut:

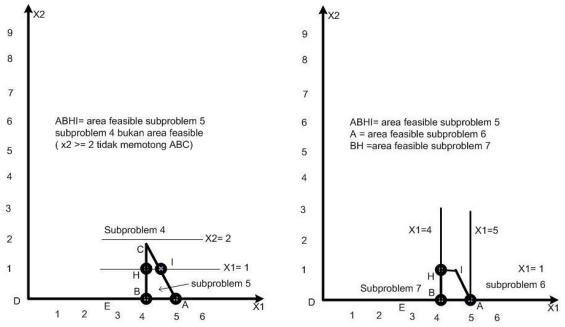
$$
\begin{aligned}
Max \quad & \mathbf{Z} = 8x_1 + 5x_2 \\
& x_1 + x_2 \leq 6 \\
& 9x_1 + 5x_2 \leq 45 \\
& x_1, x_2 \geq 0; \quad x_1, x_2, integer
\end{aligned}
$$

Contoh *Branch-and-Bound* diatas dapat digambarkan seperti pada Gambar 2, 3 dan 4.

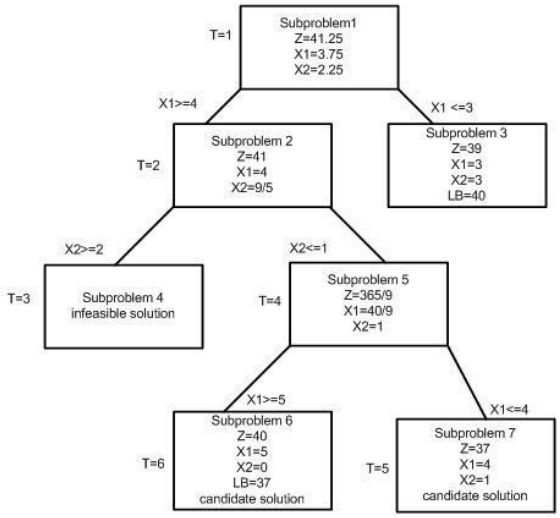
Algoritma *Branch-and-Bound* cukup efektif untuk menyelesaikan *Integer Programming*[10]. Dengan salah satu langkahnya yang tidak akan memperluas dan akan "membunuh" simpul yang tidak mungkin mengarah ke solusi, algoritma ini menjadi algoritma yang cukup efisien untuk menyelesaikan *Integer Programming*. Tetapi dalam menyelesaikan permasala-



Gambar 2. Solusi optimal linier programming subproblem 1, 2 dan 3[12]



Gambar 3. Implementasi metode *Branch-and-Bound*[12]



Gambar 4. Solusi optimal permasalahan MILP[12]

han ini, algoritma *Branch and Bound* mempunyai kelemahan, yaitu: algoritma ini tetap menghitung kemungkinan solusi dengan tipe variabel bilangan *real* walaupun pada akhirnya kemungkinan solusi ini tidak akan dipertimbangkan. Tetapi hal ini menyebabkan waktu komputasi bertambah lama.

3 *Node Hardware*

3.1 *Node Hardware Layout*

Dalam jaringan telekomunikasi terutama yang berbasis SDH (*Synchronous Digital Hierarchy*), terdiri dari beberapa perangkat:

1) *Chassis* dari perangkat (*base equipment conf*)
2) *Port Card*
3) *Pluggable device*, seperti *Gigabit Interface Converter* (GBIC), *Small Form Factor Pluggables* (SFP)
4) *Backplane power supply*
5) *Control Management*
6) *Route Processor*
7) *Memory*

3.2 Istilah pada *Node Hardware* Model

Untuk memudahkan pemahaman perangkat node telekomunikasi dalam model matematika, beberapa istilah yang dibuat untuk menggambarkan model tersebut dapat dilihat pada halaman Apendiks.

3.3 Model Matematika (MILP)

Model matematika mempunyai 2 bagian yaitu fungsi obyektif untuk menghitung minimal biaya dan batasan-batasannya[6].

$$\min_{x,y,z,\zeta} \sum_k \sum_{i \in I_k} r_{ik} x_{ik} + \sum_k t_k y_k + \sum_l c_l z_l + \bar{c}\zeta = 1 \quad (1)$$

Fungsi obyektif (1) akan menghitung biaya minimal dari *pluggable device*, *port card* dan *base equipment* berdasarkan variabel vektor (x, y, z, ζ). Persamaan diatas memiliki batasan-batasan sebagai berikut:

$$\sum_{k \in K_{ij}} x_{ik} = b_{ij}, \forall j, \forall i \in I_j \quad (2)$$

$$\sum_{i \in I_k} x_k \leq \eta_k y_k, \forall k \quad (3)$$

$$\sum_k \alpha_{hk} y_k \leq \sum_l \beta_{hl} z_l + \bar{\beta}_h \zeta, \forall h \quad (4)$$

$$z_l \leq 1 \quad (5)$$

$$x_{ik} \in N, \forall i, \forall k \quad (6)$$

$$y_k \leq 1, \forall k \quad (7)$$

$$z_l \in \{0, 1\}, \forall l_i \quad (8)$$

Penjelasan dari batasan-batasan diatas adalah sebagai berikut:

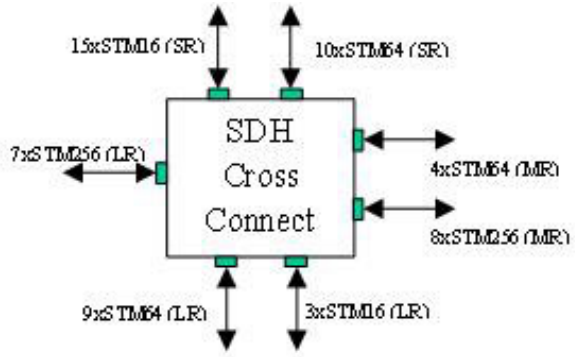
- Persamaan (2)
  b_{ij} merupakan *pluggable device* i untuk port tipe j yang telah di *install*, bij merupakan jumlah *pluggable* yang di butuhkan berdasarkan *demand matrix*. Kebutuhan ini kemudian akan di modelkan oleh variabel x_{ik} dimana k adalah pilihan dari *portcard type*.
- Persamaan (3)
  Jumlah *port* pada tiap-tiap port card yang disediakan untuk *pluggable device*. Jadi *pluggable* yang di*install* pada persamaan sisi kiri harus lebih rendah atau sama dengan bilangan jumlah *port* yang disediakan oleh *port card* pada persamaan sisi kanan.
- Persamaan (4)
  Pada sisi kiri merupakan penjumlahan *slot* yang di sediakan *port card* sedangkan sisi kanan merupakan jumlah *slot* yang disediakan oleh BEC. Sehingga jumlah keseluruhan *slot* pada *port card* harus lebih rendah atau sama dengan jumlah *slot* yang disediakan BEC.
- Persamaan (5)
  Adalah sebuah pilihan yang digunakan untuk mengaktifkan konfigurasi BEC,jika $z_l = 1$ berarti jumlah *slot* yang disediakan digunakan dalam perhitungan,akan tetapi jika $z_l = 0$ maka tidak ada konfigurasi BEC yang digunakan atau tidak *slot* yang digunakan.
- Persamaan (6)
  Memastikan bahwa variabel bilangan dari *pluggable device* adalah *integer*.
- Persamaan (7)
  Memastikan variabel bilangan *integer* dari *port card* yang ter*install*.

- Persamaan (8)
  Pilihan apakah menggunakan konfigurasi BEC atau tidak.

Penyelesaian permasalahan *integer programming* diatas akan dilakukan dengan MILP, dengan menggunakan LPSOLVE.

3.4 Data Input Sistem

Berikut adalah contoh data dari aliran trafik di sebuah *node* jaringan SDH.



Gambar 5. Data dari aliran trafik sebuah *node*

Tabel 1
Matriks Aliran Data

	STM16	STM64	STM256
Short Range	15	10	0
Medium Range	15	10	0
Long Range	15	10	0

Dan contoh untuk masukkan data adalah sebagai berikut:

[TipePort]
Nama
STM16
STM64
STM256

[PluggableDevice]
Nama
ShortRange
MediumRange
LongRange

[TipeSumberDaya]
Nama
CapSlot
CapBitrate
CapMngment

[TipePortCard]

Nama	Tipeport	JumlahPort	Harga	Capslot	CapBitrate	CapManagement
8xSTM16	STM16	8	60.5	1	20	1
16xSTM16	STM16	16	100.5	2	40	1
8xSTM16	STM16	8	220.5	2	80	1
8xSTM16	STM16	8	820.5	2	320	1

[PluggableDeviceKindsdariTipePortCard]

Jarak	TipePortCArd	Harga
ShortRange	8xSTM16	10.5
LongRange	8xSTM16	20.5
ShortRange	8xSTM64	36.5
MediumRange	8xSTM64	60.5
LongRange	8xSTM64	70.5
MediumRange	8xSTM256	260.5
LongRange	8xSTM256	310.5

[BaseEquipmentConfiguration]

Nama	Biaya	CapSlot	CapBitrate	CapMngmn
SingleChassisRouter	320.5	16	1280	62
TwoChassisRouter	800.5	30	2560	24
ThreeChassisRouter	1400.5	42	3840	36

Pada bagian ini dilakukan perencanaan implementasi program menggunakan C++ dan mengintegrasikan ke dalam lpsolve C/C++.

3.5 Metode *Heuristic*

Sebagaimana diketahui, MILP memiliki kelemahan jika data yang diolah besar dan kompleks, sehingga digunakan metode *heuristic* untuk mengatasi permasalahan tersebut. Ide dasar dari metode *heuristic* ini dari metode kompleks oleh M. J. Box[11].

Metode *heuristic* terdiri dari 3 bagian, yaitu:

1) Step 1 (Proses seleksi)
   Ini akan memilih nilai terendah dari kombinasi *pluggable* dan *port card*.
2) Step 2 (Pembentukan nilai awal)
   Berdasarkan dari data diatas (1), akan dihitung nilai dari fungsi obyektif.
3) Step 3 (Prosedur untuk mendapatkan nilai yang mendekati optimal).

Algoritma diatas akan mencari solusi yang mendekati nilai optimal berdasarkan 4 parameter yaitu *pluggable, port card, resource commodities* dan *base equipment configuration*[7].

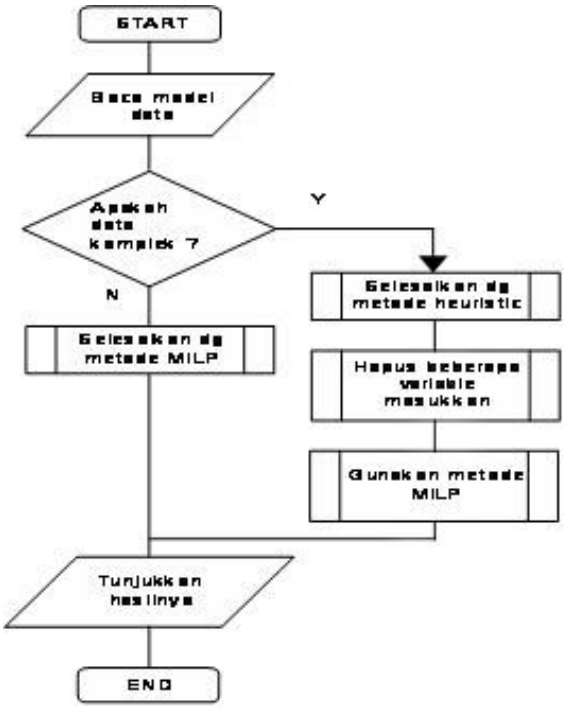
3.6 Penggabungan MILP dan Heuristic

Sesuai dengan tujuan dari penelitian ini adalah untuk mendapatkan solusi yang tepat dan cepat dalam menangani permasalahan baik yang berskala kecil maupun besar. Algoritma yang digunakan dapat dilihat pada Gambar 9.

Dari hasil eksperimen pada [7] didapatkan bahwa:

1) Waktu yang diperlukan oleh metode MILP akan meningkat secara eksponensial jika data semakin kompleks, sementara maksimum waktu yang diperlukan oleh *heuristic* adalah 0.01 detik.
2) *Mean percentage error* dari metode *heuristic* dibandingkan dengan MILP adalah dibawah 10 %.

Metode *heuristic* digunakan untuk mendapatkan solusi awal yang akan menghapus beberapa variabel BEC sebelum dimasukkan ke metode MILP. Kondisi tersebut akan menurunkan jumlah *branch-and-bounds* dari masalah yang akan dipecahkan.



Gambar 6. Penggabungan MILP dan *Heuristic*

Algoritma untuk menghapus beberapa variabel BEC sebagai berikut:

1) Baca data dari model file
2) Selesaikan dengan metode *heuristic*
3) *Output* dari *heuristic* adalah nilai awal dari nilai obyektif.
4) *Update data* dari BEC dengan menghapus beberapa nilai BEC jika biaya dari BEC lebih dari hasil No 3.
5) Selesaikan problem dengan metode MILP (LPSOLVE).
6) Tunjukkan hasilnya.

4 Hasil Eksperimen

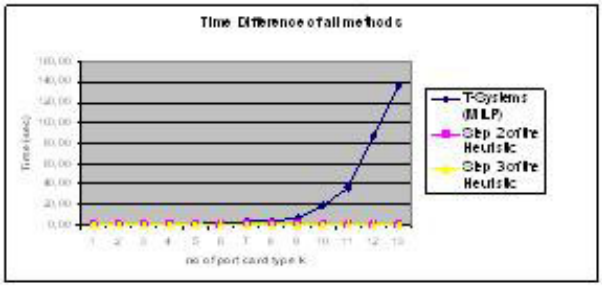
Dalam penelitian ini, penulisan program dilakukan dengan C++, dan eksperimen didasarkan pada beberapa kondisi, yaitu :

- Membandingkan hasil antara MILP dan metode *Heuristic*
- Menggabungkan metode *heuristic* dengan MILP untuk menghapus beberapa variabel di MILP.

4.1 Perbandingan Metode MILP dan *Heuristic*

4.1.1 Peningkatan Jumlah Port Card Types k

Pada pengujian ini, *port card types* akan ditingkatkan jumlahnya, dan dari gambar dibawah, semakin banyak kemungkinan dari *port card types*, maka metode MILP akan membutuhkan semakin banyak waktu komputasi secara eksponensial.



Gambar 7. Perbedaan waktu dari 2 metode untuk peningkatan jumlah *Port Card Types* k

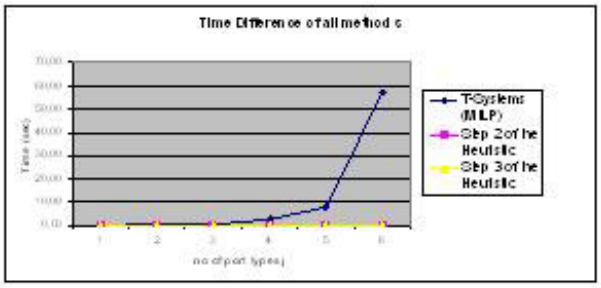
Perbandingan dari kedua metode adalah sebagai berikut :

1) Untuk metode MILP, pada data yang ke 8, waktu komputasinya akan cenderung semakin eksponensial.
2) Metode *heuristic*, ketika data semakin banyak (sampai 13 data), waktu komputasi cenderung stabil sekitar 0,01 detik.

Sehingga dalam hal ini metode heuristic terlihat jauh lebih stabil dalam hal komputasinya dibandingkan dengan metode MILP.

4.1.2 Peningkatan Jumlah Port Types j

Pada pengujian ini, *port types* akan ditingkatkan jumlahnya, dan dari gambar dibawah, semakin banyak kemungkinan dari *port types*, maka metode MILP akan membutuhkan semakin banyak waktu komputasi secara eksponensial.



Gambar 8. Perbedaan waktu dari 2 metode untuk peningkatan jumlah *Port Card Types* k

Perbandingan dari kedua metode adalah sebagai berikut:

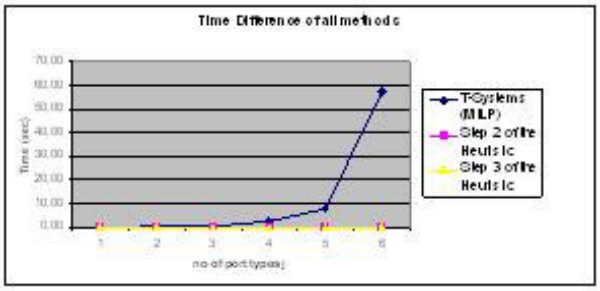
1) Untuk metode MILP, pada data yang ke 4, waktu komputasinya akan cenderung semakin eksponensial.
2) Metode *heuristic*, ketika data semakin banyak (sampai 6 data), waktu komputasi cenderung stabil sekitar 0,01 detik.

4.1.3 Peningkatan Jumlah Pluggable Device Kind i

Pada pengujian ini, *pluggable device kind* akan ditingkatkan jumlahnya, dan dari gambar dibawah, semakin banyak kemungkinan dari *pluggable device kind*, maka metode MILP akan membutuhkan semakin banyak waktu komputasi secara eksponensial.

Perbandingan dari kedua metode adalah sebagai berikut :

1) Untuk metode MILP, pada data yang ke 4, waktu komputasinya akan cenderung semakin eksponensial.
2) Metode *heuristic*, ketika data semakin banyak (sampai 6 data), waktu komputasi cenderung stabil sekitar 0,01 detik.



Gambar 9. Perbedaan waktu dari 2 metode untuk peningkatan jumlah *Pluggable Device* i

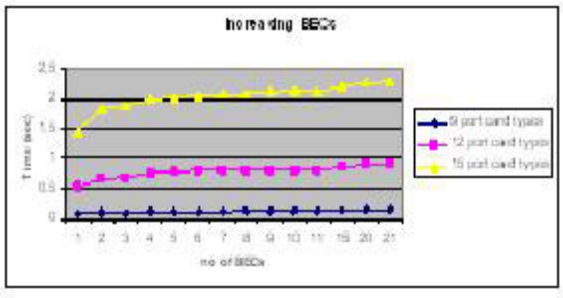
Pada pengujian ini, terbukti bahwa metode *heuristic* sangat cepat dalam melakukan kalkulasi baik ketika jumlah data kecil maupun banyak. Sedangkan metode MILP memerlukan waktu komputasi yang cenderung eksponensial ketika datanya semakin kompleks (banyak). Sehingga metode *heuristic* tersebut bisa digabungkan dengan metode MILP untuk mendapatkan waktu komputasi yang cepat dan hasil optimasi yang benar-benar optimal.

4.2 Penggabungan Metode MILP dan *Heuristic*

Pengujian dilakukan dengan cara:

1) Peningkatan jumlah *port card* sebanyak 9 buah
2) Peningkatan jumlah *port card* sebanyak 12 buah
3) Peningkatan jumlah *port card* sebanyak 15 buah

Sebelumnya akan dilakukan pengujian waktu yang diperlukan ketika jumlah BEC meningkat dengan tiga kondisi diatas.

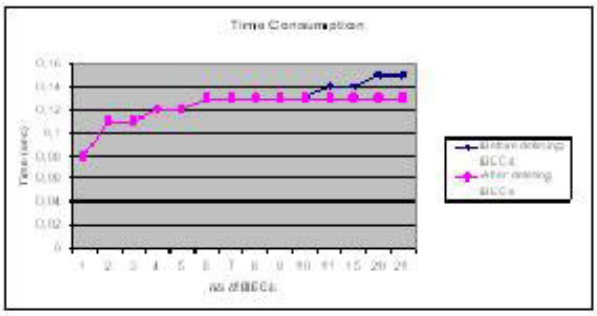


Gambar 10. Konsumsi waktu untuk peningkatan jumlah BEC

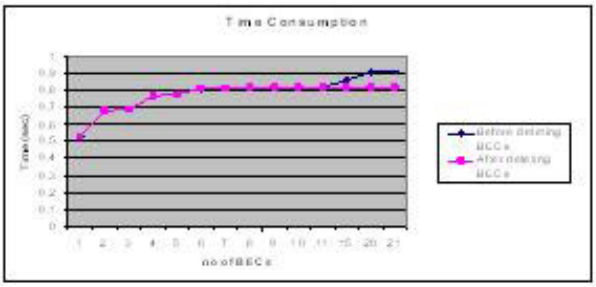
Dari Gambar 10 terlihat bahwa semakin meningkat jumlah *port card type* dan BEC maka waktu komputasi juga akan semakin meningkat.

Selanjutnya adalah pengujian dengan penggabungan metode MILP dan *heuristic*. Pengujian ditekankan pada peningkatan jumlah *port card* sebanyak 9, 12 dan 15 buah. Hasilnya dapat dilihat pada Gambar 11, 12, dan Gambar 13.

Tujuan dari tahapan ini adalah untuk menguji apakah metode *heuristic* dapat secara efektif lebih cepat ketika data semakin kompleks dengan meningkatkan jumlah *port card type k* sebanyak 9, 12 dan 15.



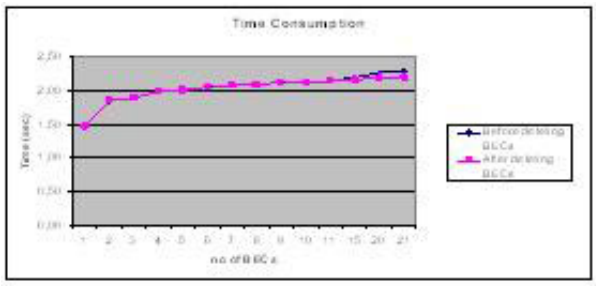
Gambar 11. Perbedaan waktu sebelum dan sesudah penghapusan BEC untuk 9 *Port Card*



Gambar 12. Perbedaan waktu sebelum dan sesudah penghapusan BEC untuk 12 *Port Card*

Pada pengujian ini dapat dianalisa bahwa perbandingan dari kedua metode adalah sebagai berikut :

1) Untuk metode MILP, pada data yang ke 4, waktu komputasinya akan cenderung semakin eksponensial, sedangkan pada metode *heuristic*, ketika data semakin banyak (sampai data ke-6), waktu komputasi masih cenderung stabil sekitar 0,01 detik.
2) Kategori data kompleks untuk menentukan kapan menggunakan penggabungan metode *heuristic* dan MILP adalah pada data ke 10.
3) Data BEC akan dihapus BEC jika harga BEC melebihi solusi dari hasil metode *heuristic*. Dan dari data diatas, terlihat bahwa harga BEC melebihi hasil *heuristic* mulai data ke 10 (seperti penjelasan pada no 2), sehingga pada data tersebut data BEC akan mulai dihapus.
4) Penambahan jumlah BEC, penggabungan 2 metode diatas masih efektif untuk menghapus beberapa data BEC dan terbukti menurunkan konsumsi waktu. Pada 9 *port card*, waktu maksimal adalah 0,13 detik. Pada 12 *port card*, waktu maksimal adalah 0,82 detik. Pada 15 *port card*, waktu maksimal adalah 2,2 detik.



Gambar 13. Perbedaan waktu sebelum dan sesudah penghapusan BEC untuk 15 *Port Card*

5 KESIMPULAN

Dari hasil eksperimen diatas dapat disimpulkan bahwa:

1) Metode *heuristic* sangat efektif untuk mendapatkan solusi awal dan sangat cepat dalam komputasinya (sekitar 0,01 detik), sedangkan MILP ketika datanya semakin kompleks (data ke 10) maka waktu komputasinya cenderung eksponensial.
2) BEC akan dihapus jika biaya dari BEC melebihi nilai awal dari hasil *heuristic*.
3) Pada peningkatan jumlah dari BEC, algoritma yang digunakan masih efektif untuk menghapus beberapa data BEC

sehingga menurunkan waktu perhitungan untuk menyelesaikan permasalahan pendimensian *node hardware*.

APENDIKS A
NODE HARDWARE

1) *Base equipment*
   Sebagai *cassing* dari *node hardware* yang terdiri dari rak, *power supply*, *controller card*, *backplane hardware*, dan lain-lain.
2) *Base equipment configuration* (BEC)
   Port card yang terinstall pada *base equipment* membutuhkan beberapa *resource* seperti *bit rate*, *slot*, dan lain-lain.
3) *Base equipment configuration* (BEC)
   Port card yang terinstall pada *base equipment* membutuhkan beberapa *resource* seperti *bit rate*, *slot*, dan lain-lain.
4) Tipe *port card*
   Merupakan *plug-in modul* yang terdiri dari satu atau beberapa port.
5) Tipe *port*
   Mewakili *port* yang ada pada *port card*. Ini berhubungan dengan *bit rate* yang berbeda-beda, seperti 1Gb/s, 2.5Gb/s.
6) *Pluggable device*
   Seperti GBIC atau SFP.

Gambar 14. Base equipment dari SDH[6]

Gambar 15. *Port card* dari beberapa vendor[6]

Gambar 16. Pluggable device dari SFP[6]

APENDIKS B
PARAMETER INPUT

Berikut adalah beberapa parameter input:

$\aleph_0$	kumpulan angka integer positive termasuk 0.
N	kumpulan angka integer positive tidak termasuk 0.
j	kumpulan index berbagai tipe port
j	port jenis $j \in J$.
K	kumpulan indek dari berbagai jenis port card.
k	indek $k \in K$ dari sebuah jenis port card.
I	kumpulan indek dari berbagai jenis *pluggable device*.
i	indek $i \in I$ dari sebuah jenis *pluggable device*.
$j(k)$	port tipe j yang disediakan oleh port card tipe k.
$\eta\kappa$	sejumlah port-port $\in N$ yang terletak pada sebuah port card bertipe k.
Ik	kumpulan indek dari *pluggable device* yang digunakan pada port card bertipe k.
Ij	kumpulan indek dari *pluggable device* untuk port bertipe j yang termasuk kumpulan index lk, dimana $j = j(k)$.
K_{ij}	Kumpulan indek dari berbagai macam tipe port card yang akan menyediakan port-port bertipe j di kombinasikan dengan *pluggable device* tipe I, dimana $j = j(k)$ dan $i \in lk$ untuk tiap k.
L	kumpulan indek dari base equipment configuration.
ℓ	indek $\ell \in L$ dari base equipment configuration.
H	kumpulan indek dari sumber barang yang diperlukan.
h	indek $h \in H$ dari *resource type* yang diperlukan.
αhk	syarat kapasitas ≥ 0 dari *resource* yang diperlukan h untuk sebuah port card tipe k.
βhk	kapasitas dari konfigurasi base equipment didalam unit αhk
Γ_{ik}	harga ≥ 0 dari *pluggable device* jenis I dengan port card k.
t_K	harga ≥ 0 dari port card k dari beberapa *base equipment configuration*
C_ℓ	harga tetap ≥ 0 dari base equipment configuration
$b_{ij}j$	bilangan N dari *pluggable device* i yang sesuai dengan port tipe j yang menghubungan aliran data sampai ke node

APENDIKS C
PARAMETER OUTPUT

Parameter output harus di hitung:

- X_{ik} *pluggable device* i yang di install pada *port card* tipe k
- Y_k banyaknya *port card* tipe k yang di install.
- Z_ℓ binary yang menentukan variabel mana yang mengindikasikan apakah konfigurasi *base equipment* sudah di install.

UCAPAN TERIMAKASIH

Paper ini didanai oleh UPPM PENS-ITS dalam rangka penelitian lokal tahun 2009.

DAFTAR PUSTAKA

[1] Alexander Gersht, Robert Weihmayer, *A Mixed Integer/Linear Programming Approach to Communication Network Design*, IEEE Proceedings of 26th Conference on Decision and Control, Athens, Greece, December 1986.

[2] Song Guo, Oliver Yang, *Minimum-Energy Multicast in Wireless Ad Hoc Networks with Adaptive Antennas: MILP Formulations and Heuristic Algorithms*, IEEE Transactions on Mobile Computing, Vol 5, No. 4, April 2006.

[3] Alexander Gerhst, Robert Weihmayer,*Joint Optimization of Data Network Design and Facility Selection*, IEEE Journal on Selected Areas in Communications, Vol. 8, No. 9, December 1990.

[4] Ashwin Gumaste, Paparao Palacharla, *Heuristic and Optimal Techniques for Light-trail Assignment in Optical Ring WDM Networks*, Proceeding of Computer Communications, Volume 30, pages 990-998, March 2007.

[5] LTM Berry, S. Khler, D Staehle and P Tran-Gia, *Fast heuristics for optimal routing in large IP networks*,Universitt Wrzburg Germany, Institut fr Informatik Research Report Series, Report No. 262 July 2000.

[6] Koerkel, M. F., *Node Layout Model for Telecommunication Networks*, T-Systems Enterprise Services GmbH, Version: 0.99d, June 19, 2008.

[7] Hadi, M. Z., *A Node Hardware Dimensioning Model For Telecom Networks using Heuristic Method*, Seminar IES 2008, PENS-ITS

[8] Telecommunication Network, NCS Communications Engineering, Pte.Ltd. http://www.ncs.com.sg/documents/One-time-Telco-Network -041203- final.pdf

[9] Marconi Communications GmbH, 2000,*Introduction to the Synchronous Digital Hierarchy* - SDH Basics

[10] Shieny aprilia, *Aplikasi Algoritma Branch and Bound untuk menyelesaikan integer programming*, http://www.informatika.org/ rinaldi/Stmik/2006-2007/Makalah 2007 /MakalahSTMIK2007-076.pdf ,desember 3,2008

[11] Box, M. J., *A new method of constrained optimisation and comparison with other methods*, Computer Journal, Volume 7, pages 42 - 52, 1965

[12] Kalyanmoy Deb,*Optimization for Engineering Design: Algorithms and Examples*, Prentice Hall of India Pvt. Ltd.,Feb 2004

[13] Reeves,*Modern Heuristic Techniques for Combinatorial Problems*, Publisher: Halsted Press (April 1993)

[14] Ray Hill, *Tabu Search For Military Analysis*, Department of Operational Sciences Air Force Institute of Technology, 2002

[15] Gabriela Voicu, *A Comparative Analysis of Traveling Salesman Heuristic Implementations in GIS*, GIS Masters Project - Summer 2006

M.Zen Samsono Hadi lahir di Kediri, ia memperoleh gelar Sarjana Teknik (ST) pada Jurusan Teknik Elektro pada tahun 2000 dan magister teknik (MT) pada tahun 2009, keduanya dari Institut Teknologi Sepuluh Nopember (ITS) Surabaya. Pada tahun 2007, ia mendapat kesempatan untuk program double degree ke Jerman, dan mendapatkan gelar master of science pada tahun 2008. Ia adalah pengajar pada jurusan Teknik Telekomunikasi, Politeknik Elektronika Negeri Surabaya. Bidang penelitian yang ditekuni adalah *network security*, *network design* dan *internet application*. Pernah melakukan penelitian pada bidang network design di *T-Systems Enterprise* GmbH, Darmstadt Jerman pada tahun 2008.

Aries Pratiarso lahir di Surabaya, ia memperoleh gelar Sarjana Teknik (ST) pada Jurusan Teknik Elektro pada tahun 1994 dan Magister Teknik (MT) pada tahun 2004, keduanya dari Institut Teknologi Sepuluh Nopember (ITS) Surabaya. Ia adalah pengajar pada jurusan Teknik Telekomunikasi, Politeknik Elektronika Negeri Surabaya. Bidang penelitian yang ditekuni adalah *Wireless Communication*, teknik koding dan *image processing*.

Muhammad Agus Zainuddin lahir di Surabaya pada tanggal 12 Agustus 1978. Menyelesaikan pendidikan strata satu di Jurusan Teknik Elektro, Fakultas Teknik, Universitas Brawijaya Malang, lulus tahun 2003 mendapat gelar Sarjana Teknik dan mendapat gelar Magister Teknik pada tahun 2007 dari Institut Teknologi Sepuluh Nopember (ITS) Surabaya. Ia adalah pengajar pada jurusan Multimedia Broadcasting, Politeknik Elektronika Negeri Surabaya. Bidang penelitian yang ditekuni adalah *data compression* dan *error correction*.