PENGEMBANGAN MODEL REPRESENTASI KONTINU UNTUK PROBLEM-PROBLEM TATA LETAK FASILITAS DENGAN LUAS TIDAK SAMA MENGGUNAKAN ALGORITMA DIFFERENTIAL EVOLUTION

SKRIPSI

STEVEN SULISTYO OETOMO
0606077586

UNIVERSITAS INDONESIA
FAKULTAS TEKNIK
PROGRAM STUDI TEKNIK INDUSTRI
DEPOK
JUNI 2010

PENGEMBANGAN MODEL REPRESENTASI KONTINU UNTUK PROBLEM-PROBLEM TATA LETAK FASILITAS DENGAN LUAS TIDAK SAMA MENGGUNAKAN ALGORITMA DIFFERENTIAL EVOLUTION

SKRIPSI

Diajukan sebagai salah satu syarat untuk memperoleh gelar Sarjana Teknik

STEVEN SULISTYO OETOMO
0606077586

UNIVERSITAS INDONESIA
FAKULTAS TEKNIK
PROGRAM STUDI TEKNIK INDUSTRI
DEPOK
JUNI 2010

HALAMAN PERNYATAAN ORISINALITAS

Skripsi ini adalah benar hasil karya saya sendiri,
dan semua sumber baik yang dikutip maupun yang dirujuk
telah saya nyatakan dengan benar

Nama	**: Steven Sulistyo Oetomo**
NPM	**: 0606077586**
Tanda tangan	**:**
Tanggal	**: Juni 2010**

HALAMAN PENGESAHAN

Skripsi ini diajukan oleh

Nama : Steven Sulistyo Oetomo
NPM : 0606077586
Program Studi : Teknik Industri
Judul Skripsi : Pengembangan Model Representasi Kontinu Untuk Problem-problem Tata Letak Fasilitas Menggunakan Algoritma Differential Evolution

Telah berhasil dipertahankan di hadapan dewan penguji sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Sarjana Teknik pada Program Studi Teknik Industri, Fakultas Teknik, Universitas Indonesia.

DEWAN PENGUJI

Pembimbing : Armand Omar Moeis, S.T. ,M.Sc (.....)

Penguji : Arian Dhini, S.T., M.T. (.....)

Penguji : Ir Amar Rachman, MEIM. (.....)

Penguji : Ir. Yadrifil, M.Sc. (.....)

Ditetapkan di : Depok
Tanggal : 30 Juni 2010

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI
TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

Sebagai sivitas akademik Universitas Indonesia, saya yang bertanda tangan di bawah ini:

Nama : Steven Sulistyo Oetomo
NPM : 0606077586
Program Studi : Teknik Industri
Departemen : Teknik Industri
Fakultas : Teknik
Jenis Karya : Skripsi

Demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Indonesia **Hak Bebas Royalti Non-eksklusif (*Non-exclusive Royalty Free Right*)** atas karya ilmiah saya yang berjudul:

Pengembangan Model Representasi Kontinu Untuk Problem-problem Tata Letak Fasilitas Menggunakan Algoritma Differential Evolution

beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Non-eksklusif ini Universitas Indonesia berhak menyimpan, mengalih media/formatkan, mengelola dalam bentuk pangkalan data (*database*), merawat dan mempublikasikan tugas akhir saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di: Depok
Pada tanggal: Juni 2010
Yang menyatakan

(Steven Sulistyo Oetomo)

KATA PENGANTAR

Puji syukur kepada Tuhan Yang Maha Esa atas segala berkat dan kasih karunia-Nya penulis panjatkan, sehingga skripsi ini dapat selesai dengan baik. Penulisan skripsi ini dilakukan dalam rangka melengkapi salah satu persyaratan untuk menyelesaikan Program Pendidikan Sarjana Teknik Industri, Fakultas Teknik Universitas Indonesia. Penulis sangat berterima kasih atas segala bantuan, komentar, dan bimbingan dari berbagai pihak yang sangat berarti bagi penulis untuk menyelesaikan skripsi ini. Rasa terima kasih saya itu saya ucapkan kepada:

1) Bapak Prof. Dr. Ir. *T. Yuri* M. *Zagloel*, MEng.Sc, selaku Ketua Departemen Teknik Industri Universitas Indonesia, yang telah memberikan banyak bimbingan bagi para mahasiswanya.
2) Orang tua, (Alm) Bapak Oetomo Prabowo dan Ibu Isabela Ronni Chandra, atas segala doa, kasih sayang, nasehat, dan wujud dukungan lainnya yang telah diberikan kepada penulis dengan cuma-cuma.
3) Bapak Armand Omar Moeis, S.T. ,M.Sc, selaku dosen pembimbing skripsi, untuk segala ilmu, ide, saran, dukungan riil maupun moriil, dan pengarahannya sehingga skripsi ini dapat terwujud.
4) Bapak Komarudin, selaku dosen pembimbing skripsi kedua, atas segala ilmu pengetahuan, bimbingan, dan sarannya, sehingga penulis mendapatkan ide dasar untuk skripsi ini.
5) Bapak Dr. Rainer Storn dan Bapak Kenneth Price, atas sumbangsihnya yang amat besar yang berupa *open source code* untuk program algoritma yang penulis buat, sehingga program dapat berjalan dengan baik sesuai harapan.
6) Lucia Roly Prihandini atas pengajaran dan bimbingannya, serta kesediaannya dan kesigapannya dalam menolong sehingga penulis dapat belajar bahasa pemrograman Java dengan baik.
7) Sisca Pratiwi dan Satria Hutomo Jihan, sebagai rekan seperjuangan dalam mengerjakan skripsi ini, sekaligus sudah menjadi pendengar yang baik ketika penulis resah dan galau.
8) M.Aditia Eka Putra yang telah banyak memberikan bantuan dan membantu penulis mencari pengajar bahasa pemrograman Java.

9) Billy, Andito Murti, Irvan Ramadhan Dwi Putra, dan Fadhillah Meizi atas segala diskusi yang berbobot mengenai skripsi, sehingga menambah wawasan penulis.
10) Teman-teman Teknik Industri Universitas Indonesia angkatan 2006, atas segala suka duka, kebersamaan, dan keceriaan selama 4 tahun berkuliah bersama.
11) Semua pihak yang terlibat dan telah membantu penulis mewujudkan skripsi ini sehingga dapat selesai dengan baik.

Akhirnya, penulis berharap agar skripsi ini bisa memberikan inspirasi dan manfaat bagi semua pihak yang membacanya dan bagi pengembangan ilmu pengetahuan.

Depok, 30 Juni 2010

Penulis

ABSTRAK

Nama : Steven Sulistyo Oetomo
Program Studi : Teknik Industri
Judul : Pengembangan Model Representasi Kontinu Untuk Problem-problem Tata Letak Fasilitas Menggunakan Algoritma Differential Evolution

Hingga saat ini tidak ada satupun metode pendekatan yang ampuh diterapkan untuk mendapatkan solusi dari Unequal Area Facility Layout Problems untuk semua kasus, sehingga penelitian ini dilakukan untuk meneliti kemungkinan metode representasi kontinu dengan bantuan algoritma Differential Evolution dapat menjadi metode yang lebih baik dari metode-metode sebelumnya. Objek perbandingan yang digunakan adalah nilai rekor optimum yang dapat dicapai oleh masing-masing metode. Hasil dari penelitian ini adalah data-data nilai optimum yang dihasilkan oleh algoritma Differential Evolution untuk dua puluh kasus Unequal Area Facility Layout Problems yang sudah pernah dicari nilai optimalnya oleh metode lainnya. Setelah penelitian dilakukan, algoritma Differential Evolution mampu meningkatkan nilai optimal untuk satu dari dua puluh problem.

Kata kunci:
Optimasi, algoritma *Differential Evolution*, *Unequality Area Facility Layout Problems*

ABSTRACT

Name : Steven Sulistyo Oetomo
Study program : Industrial Engineering
Title : Developing Continue Representation Model For Unequality Area Facility Layout Problems Using Differential Evolution Algorithm

Up until now there is no single method that able to generate solution from Unequal Area Facility Layout Problems for all cases, therefor this research objective is to find the possibility of continue representation method with Differential Evolution algorithm becomes a better method than preovius methods. The comparison object is optimum value record that can be achieved by those methods. The result of this research is datas of optimum value generated by Differential Evolution algorithm for twenty cases of Unequality Area Facility Layout Problems that has been solved by the other methods. After this research done, Differential Evolution algorithm can only improve optimum value for one from twenty cases.

Key words:
Optimization, Differential Evolution algorithm, Unequality Area Facility Layout Problems

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR LAMPIRAN

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Persaingan usaha dewasa ini menuntut para pelaku bisnis untuk bersaing ketat agar dapat bertahan dalam kompetisi usaha. Hal-hal penting seperti kualitas barang dan jasa, harga, ketersediaan barang, *supply chain*, dan hal-hal lainnya sudah mendapat banyak perhatian khusus dari kebanyakan produsen. Meskipun demikian, sering kali faktor tata letak pabrik luput dari perhatian, meskipun tata letak pabrik sangat vital, karena berperan dalam kegiatan pabrik selama fasilitas pabrik tersebut masih digunakan. Dengan demikian, sebagai salah satu bentuk investasi, tata letak pabrik tidak boleh didesain dengan sembarangan.

Ketika akan mempersiapkan sebuah pabrik, salah satu hal yang akan dilakukan pertama kali adalah melakukan desain tata letak pabrik. Permasalahan yang dihadapi ketika melakukan desain tata letak pabrik adalah luas tanah yang dibeli untuk area pabrik, jumlah departemen atau stasiun kerja, dan luas masing-masing departemen; untuk mendapatkan biaya penanganan material (*material handling cost*) antar departemen/antar stasiun kerja yang seoptimal mungkin.

Perencanaan tata letak fasilitas adalah salah satu bidang ilmu yang sudah dipelajari dan dikembangkan bertahun-tahun. Problem ini amat sulit untuk dipecahkan karena konstrain-konstrain yang dimilikinya. Permasalahan yang menjadi sorotan utama adalah ketidaksamaan luas antar departemen/antar stasiun kerja (*Unequal Area Facility Layout Problem*/UA-FLP), yang mana menyebabkan jumlah kombinasi posisi stasiun kerja yang memungkinkan. Perbedaan cara penempatan stasiun kerja, baik secara lokasi maupun horizontal atau harus vertikal, mempengaruhi biaya penanganan material. Penempatan ini dilakukan tidak semata-mata berdasarkan faktor estetika semata, melainkan berdasarkan perhitungan biaya aktivitas yang akan terjadi dengan desain tata letak pabrik tertentu.

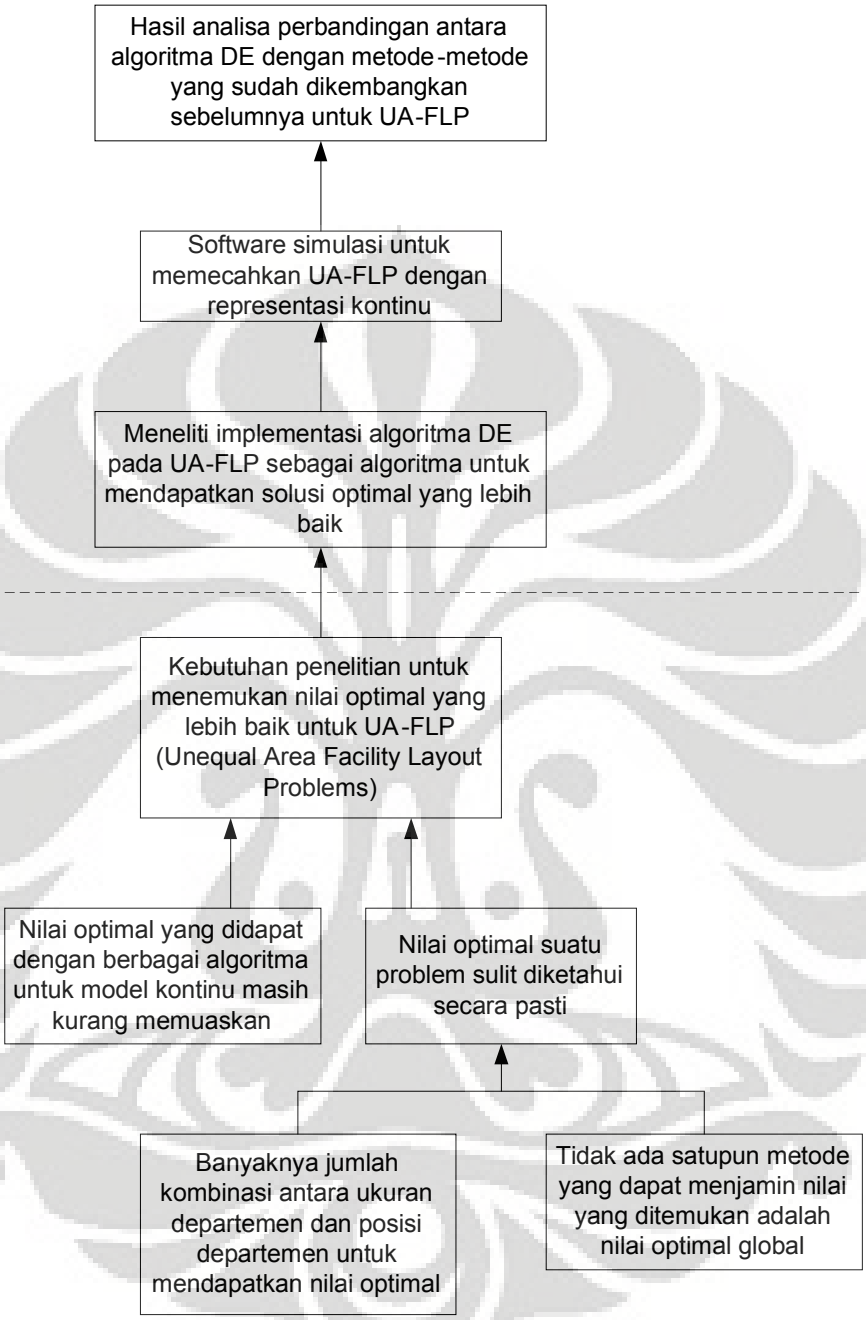
Pencarian solusi untuk UA-FLP ini hampir selalu menggunakan pendekatan-pendekatan metaheuristik. Alasan utama mengapa hampir selalu menggunakan pendekatan metaheuristik adalah karena wilayah pencarian dari

problem-problem UA-FLP memiliki begitu banyak *local optima* atau nilai optimal untuk satu area pencarian. Pendekatan-pendekatan konvensional menjadi tidak layak diimplementasikan pada UA-FLP, karena pendekatan-pendekatan tersebut tidak dapat menjamin bahwa nilai solusi yang didapat bukanlah nilai *local optima*. Dengan demikian metode metaheuristik menjadi lebih layak dipertimbangkan karena pendekatan-pendekatan metaheuristik pada umumnya memang dibuat agar tidak terjebak oleh nilai-nilai *local optima* tersebut.

Sampai dengan saat skripsi ini dibuat, metode-metode yang sudah diaplikasikan untuk mendapatkan solusi dari UA-FLP tergolong metode metaheuristik seperti: *ant colony optimization* (ACO), *tabu search,* dan lain-lain. Metode-metode tersebut menggunakan berbagai bentuk representasi, mulai dari *Quadratic Assignment Model* (QAP), *Flexible Bay Structure Model* (FBS), dan *Slicing Tree Structure Model* (STS). Sebagai gambaran, metode-metode konvensional hanya dapat mencari solusi optimal hingga 11 departemen saja (Meller *et al*, 2007), padahal ukuran dari problem semakin meningkat dan problem terbesar hingga tulisan ini dibuat memiliki 35 departemen (Liu dan Meller, 2007). Pada dasarnya pendekatan-pendekatan metaheuristik tidak dapat menjamin bahwa nilai solusi yang didapat adalah nilai yang optimum. Hal ini membuka peluang untuk pendekatan-pendekatan metaheuristik lainnya untuk diterapkan pada UA-FLP ini.

Pencarian solusi dilakukan melalui simulasi yang akan diterapkan pada *Java programming*. Simulasi dilakukan pada *Java* karena karakter *object oriented* pada *Java programming* dapat memudahkan pengembangan metode ini lebih lanjut, dan memudahkan dalam pembuatannya. Lagipula *Java* memiliki fasilitas bantuan *syntax* online berupa *library online* untuk menyimpan banyak rumus-rumus kode yang sudah pernah dibuat sebelumnya oleh pembuat program yang lain. Dengan mengimplementasikan metode DE ini pada *Java*, dibuka kesempatan untuk peneliti lain untuk melakukan perbaikan ataupun pengembangan untuk metode ini.

1.2 Diagram Keterkaitan Masalah



Gambar 1.1 Diagram Keterkaitan Masalah

1.3 Perumusan Masalah

Berdasarkan diagram keterkaitan masalah pada gambar 1.1 di atas, permasalahan yang akan diteliti lebih lanjut pada skripsi ini adalah pengembangan

metode baru representasi kontinu yang dapat digunakan untuk memecahkan masalah UA-FLP.

1.4 Tujuan Penelitian

Tujuan penelitian ini adalah sebagai berikut:

- Mengaplikasikan algoritma DE untuk mencari solusi dari problem UA-FLP
- Meneliti kelebihan dan kekurangan algoritma DE dibandingkan dengan algoritma-algoritma yang sudah pernah diaplikasikan UA-FLP sebelumnya, dalam hal nilai optimal yang dihasilkan saja.
- Meneliti perilaku algoritma DE pada UA-FLP.
- Mendapatkan nilai solusi optimal yang baru untuk UA-FLP.

1.5 Batasan Penelitian

Penelitian ini memiliki batasan sebagai berikut:

- Penelitian ini berfokus pada desain tata letak pabrik yang wilayahnya tidak rata (unequal area).
- Kasus yang digunakan sebagai perbandingan kinerja metode sebatas pada dua puluh kasus pada dua puluh kasus yang ada pada **Lampiran 1**.
- Penelitian hanya meneliti kinerja algoritma DE pada F = 0.8 dan CR = 0.8.
- Penelitian hanya meninjau nilai fungsi biaya total sebagai hasil dari algoritma DE yang diimplementasikan pada UA-FLP
- Pembuatan model dilakukan dengan program komputer berorientasi objek untuk memudahkan penelitian

1.6 Metodologi Penelitian

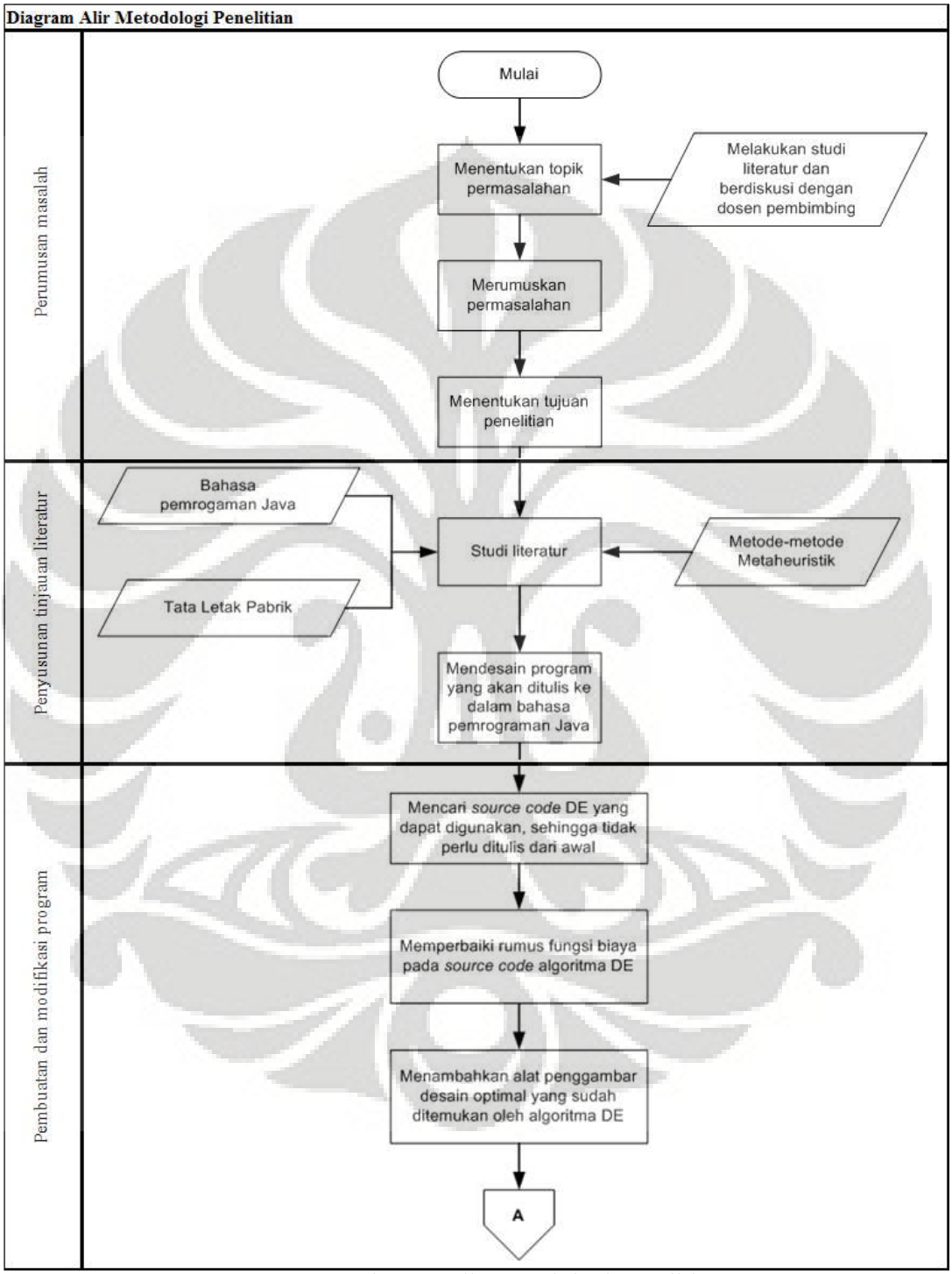
Penelitian ini dilakukan dengan tahap-tahap sebagai berikut:

- Perumusan masalah
- Penyusunan tinjauan literatur
- Membuat *software* simulasi
- Pengumpulan dan pengolahan data
- Analisis
- Kesimpulan dan saran

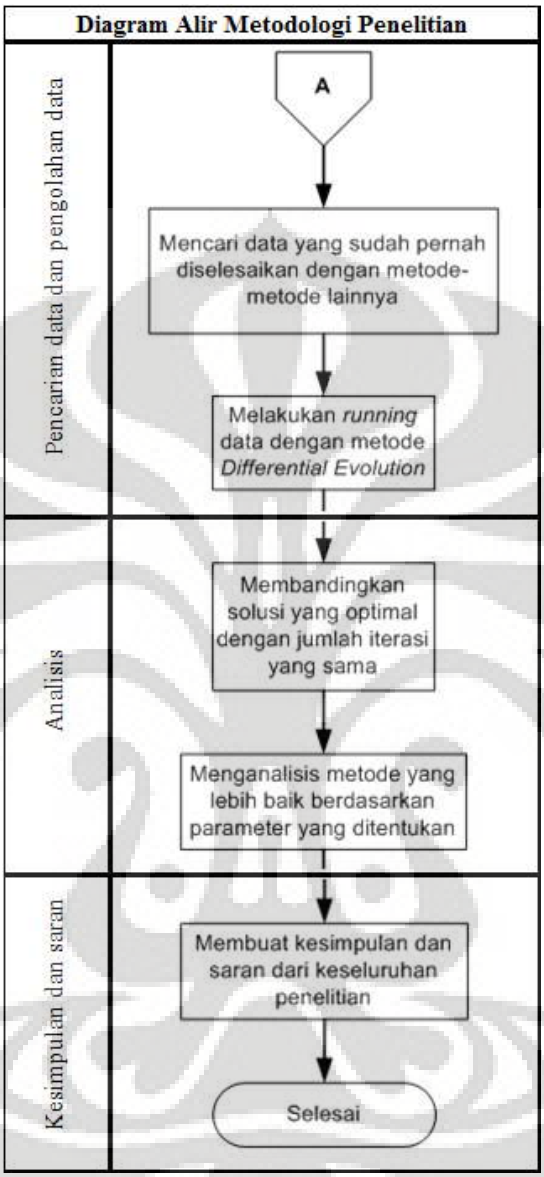
Adapun asumsi-asumsi yang diambil untuk keperluan penelitian ini adalah sebagai berikut:

1) Diasumsikan antara ruang yang satu dengan ruang yang lain dapat beririsan. Akan tetapi agar desain keluaran layak, serta nilai biaya totalnya dapat diperbandingkan, maka diterapkan suatu pinalti yang mana besarnya tidak berlebihan, sehingga nilai biaya totalnya dapat diperbandingkan dengan biaya total dari desain yang dicari dengan algoritma lainnya.
2) Diasumsikan nilai biaya penanganan material untuk setiap arus material antar departemen adalah satu satuan. Asumsi ini diambil dalam rangka penelitian, untuk menyetarakan kasus yang satu dengan kasus lainnya sekaligus menyederhanakannya, sehingga perbedaan yang timbul hanyalah besar arus material dan departemen yang terlibat saja.
3) Diasumsikan berapapun nilai parameter F dan CR pasti akan menghasilkan nilai optimal yang sama.

1.7 Diagram Alir Metodologi Penelitian



Gambar 1.2 Diagram alir metodologi penelitian



Gambar 1.2 Diagram alir metodologi penelitian (sambungan)

1.8 Sistematika Penulisan

Penulisan skripsi ini dibagi menjadi lima bab, yaitu:

Bab 1 adalah bab pendahuluan. Bab ini berisikan latar belakang permasalahan, diagram keterkaitan masalah, perumusan masalah, tujuan penelitian, ruang lingkup penelitian, metodologi penelitian, dan sistematika penulisan.

Bab 2 adalah dasar teori. Bab ini membahas teori-teori mengenai pengerjaan problem UA-FLP dengan algoritma *Ant Colony Optimization*, algoritma *Differential Evolution*, pemilihan parameter F dan CR untuk pencarian, dan algoritma yang digunakan untuk simulasi. Dibahas pula mengenai *facility layout planning*, mengenai beberapa representasi yang sudah pernah dikembangkan untuk mendapatkan solusi UA-FLP.

Bab 3 berjudul pengumpulan data. Pengumpulan data dan detail daftar data-data yang digunakan untuk bahan penelitian dibahas pada bab ini, sedangkan detail dari setiap problem yang digunakan cukup ditempatkan pada bagian lampiran, karena data-data tersebut merupakan data-data sekunder.

Bab 4 berjudul analisis. Perencanaan dan penulisan program akan dibahas pada bab ini, meliputi penjelasan rumus-rumus yang dipakai pada fungsi evaluasi biaya total. Setelah pembahasan program, akan diuji kelayakan dari program yang mana menguji konsistensi dari program dalam menghasilkan nilai-nilai optimal dengan menggunakan *reliability analysis*. Setelah uji *reliability analysis* peneliti akan menganalisis hasil yang didapat dari program algoritma DE dibandingkan dengan hasil *reliability analysis* , dilanjutkan dengan pembandingan dengan nilai-nilai terbaik yang diketahui oleh literatur-literatur lainnya jika terbukti konsisten.

Bab 5 adalah bab untuk kesimpulan dan saran. Merupakan bab terakhir yang akan membahas kesimpulan dan saran dari penulis. Kesimpulan ini sesuai dengan penelitian yang sudah penulis lakukan, sedangkan saran merupakan opini pribadi penulis untuk penelitian lebih lanjut yang mungkin akan dilakukan.

BAB 2
DASAR TEORI

Bab ini akan menerangkan mengenai tinjauan pustaka atau dasar teori dari skripsi. Untuk itu, bab ini akan terbagi menjadi 3 sub bab, yakni sub bab UA-FLP (*Unequality Area Facility Layout* Problem), sub bab DE (*Differential Evolution*), sub bab program model penyelesaian dengan menggunakan bantuan *Java Programming Language*, dan sub bab tambahan yang akan menjelaskan keterkaitan antara algoritma DE, UA-FLP, dan program model penyelesaian.

2.1 UA-FLP (*Unequality Area Facility Layout Problem*)

Yang tergolong problem tata letak pabrik (FLP) adalah segala hal yang berkaitan dalam menentukan wujud fisik organisasi dari sistem produksi. Kata kunci 'sistem produksi' tidak hanya berlaku pada industri manufaktur saja, tetapi juga berlaku pada industri jasa dan sistem komunikasi (Meller and Gau, 1996). Tujuan utama dari memecahkan kasus FLP adalah untuk meminimalkan biaya penanganan material total dari sebuah sistem produksi. Tujuan ini juga dapat mencakup atau pun dapat dikombinasikan dengan berbagai tujuan lainnya seperti memaksimalkan pendayagunaan ruang, memaksimalkan fleksibilitas, dan memaksimalkan kepuasan karyawan dan keselamatan (Muther, 1955).

Tata letak pabrik berperan penting dalam lingkungan manufaktur, karena tata letak pabrik menentukan hubungan fisik antar aktivitas manufaktur. Tata letak pabrik merepresentasikan lokasi-lokasi dari setiap sub fasilitas yang mana material bergerak dari satu tempat ke tempat lain. Dengan demikian, tata letak pabrik menentukan jarak tempuh material pada proses manufaktur. Oleh sebab itu, tata letak pabrik dapat dianggap sebagai sebuah pendekatan untuk mengontrol biaya pergerakan material. Pada industri manufaktur, biaya pergerakan material terhitung sebagai bagian dari biaya penanganan material. Penurunan biaya pergerakan material akan menurunkan biaya penanganan material total, dan pada akhirnya akan menurunkan biaya manufaktur.

FLP adalah salah satu probem pada perekayasaan dan masih menjadi area yang aktif hingga saat ini. Muther (1955) berhasil mengembangkan sebuah prosedur

sistematis untuk mendesain tata letak pabrik pada 1950-an. Penelitian pada tata letak pabrik masih berlanjut hingga saat ini, meliputi inovasi dan pengembangan untuk mengintegrasikan semua komponen dalam perencanaan fasilitas.

Muller (1955) sudah membuat daftar tujuan dari tata letak pabrik. Meskipun tujuan-tujuan ini ditujukan untuk tata letak pabrik, tujuan-tujuan ini juga relevan dengan tata letak fasilitas. Tujuan-tujuan tersebut adalah sebagai berikut:

1) Untuk menekan jarak perjalanan dari material-material.
2) Untuk mendapatkan alur regular dari material-material dan produk dan untuk menghilangkan kemacetan pada produksi.
3) Untuk mengefektifkan pendayagunaan ruang yang digunakan oleh fasilitas-fasilitas.
4) Untuk meningkatkan kepuasan dan keselamatan pekerja.
5) Untuk mendapatkan fleksibilitas yang dapat disesuaikan dengan mudah sesuai dengan kondisi.

FLP dapat dibagi menjadi tiga kategori utama, yaitu: QAP, UA-FLP, dan MLP. Perbandingan dari ketiga kategori ini dapat dilihat pada **Tabel 2.1**.

Tabel 2.1 Perbandingan Antara Ketiga Kategori FLP

No	Tipe Problem	Department Size	Location Candidate	Facility Area
1	QAP	Ukuran sama, dimensi tetap atau diacuhkan	Lokasi tetap	Tetap atau diacuhkan
2	UA-FLP	Ukuran beragam, variabel-variabel dipilih	Variabel-variabel diputuskan	= total luas departemen
3	MLP	Ukuran beragam, dimensi tetap	Variabel-variabel diputuskan	= total luas departemen atau berdimensi bebas

QAP diperkenalkan oleh Koopmans dan Beckman (1957) untuk memodelkan problem pengalokasian fasilitas-fasilitas yang berhubungan, dengan luas yang sama. Tujuan dari QAP adalah untuk menemukan sebuah cara untuk menempatkan setiap fasilitas sehingga biaya total dari penempatan fasilitas dapat ditekan. QAP adalah kasus khusus dari FLP yang mana semua departemen memiliki luas yang sama dan semua lokasi sudah tetap dan diketahui.

UA-FLP pertama kali diformulasikan oleh Armour dan Buffa (1963). Tujuannya adalah untuk mempartisi sebuah wilayah menjadi beberapa sub wilayah departemen sehingga dapat meminimalkan biaya penanganan material total. Luas departemen-departemen tidak perlu sama, tetapi ada konstrain rasio dimensi atau batasan dimensi minimum. Para peneliti sudah mengajukan beberapa representasi untuk menekan kompleksitas dari UA-FLP. Dua representasi yang sukses antara lain adalah model *Flexible Bay Structure* (FBS) dan model *Slicing Tree Structure* (STS) (Meller dan Gau, 1996). Kedua model ini membuat para peneliti dapat menggunakan algoritma optimasi berbasis diskrit untuk memecahkan UA-FLP.

Berbeda pula dengan MLP yang memiliki beberapa mesin dengan ukuran yang tetap dan sejumlah produk yang harus diproduksi dengan proses-proses yang spesifik. Tujuannya adalah untuk menentukan lokasi dari setiap mesin pada sebuah ruang dan pada saat yang sama meminimalkan penanganan material (Tompkins, 2003).

UA-FLP terutama digunakan untuk memodelkan tata letak manufaktur. Problemnya muncul ketika ada kebutuhan untuk membuat atau memodifikasi sebuah tata letak fasilitas. Kebutuhan tersebut muncul disebabkan oleh pengembangan fasilitas manufaktur yang baru, ekspansi dari fasilitas manufaktur yang lama, perubahan jumlah produksi dan spesifikasi, dan lain sebagainya.

UA-FLP pada awalnya diformulasikan oleh Armour dan Buffa (1963). Mereka mengasumsikan bahwa ada beberapa wilayah rectangular yang tetap, dengan dimensi H dan W, yang mana H adalah tinggi (*heights*) dan W adalah lebar (*width*). Juga ada beberapa fasilitas yang butuh untuk ditempatkan pada fasilitas. Jumlah departemen, luas setiap departemen, dan nilai arus material yang berkaitan dengan setiap pasang dari departemen diasumsikan sudah diketahui. Fungsi tujuan untuk meminimalkan biaya pergerakan material total adalah sebagai berikut:

$$Total\ Cost = \sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}.f_{ij}.c_{ij} \qquad (2.1)$$

n = jumlah departemen

f_{ij} = arus material total antara departemen i dengan departemen j, yang mana i,j = 1,2,…,n

d_{ij} = jarak antara departemen i dengan departemen j (i,j = 1,2,…,n)

c_{ij} = biaya dari menggerakkan satu unit material per unit jarak dari departemen i ke j, yang mana i,j = 1,2,…,n

Di samping rumus fungsi biaya di atas, UA-FLP juga memiliki tiga konstrain yang menjadi perhatian pula. Ketiga konstrain itu adalah: 1) semua departemen harus ditempatkan di dalam wilayah fasilitas, 2) semua departemen tidak boleh saling beririsan, dan 3) dimensi-dimensi semua fasilitas harus memenuhi aturan rasio panjang lebar atau panjang lebar minimum. Sehingga untuk memenuhi konstrain tersebut, fungsi biaya tujuan berubah menjadi:

$$Total\ cost = \sum_{i=1}^{N}\sum_{j=1}^{N}(d_{ij} \times f_{ij} \times c_{ij} + IA_{ij} \times \frac{(\sum L + \sum H) \times \sum f}{L_{fac} \times H_{fac}} \times penalty\ modifier \quad (2.2)$$

Sebagai tambahan, ukuran setiap departemen pada UA-FLP biasanya memiliki konstrain tambahan agar wujud sebuah departemen tidak terlalu ramping seperti jarum. Ada dua konstrain yang dikenal untuk mencegah hal tersebut, yang pertama adalah I_{min} atau panjang lebar minimum sebuah departemen; dan *Aspect Ratio* atau rasio maksimal untuk perbandingan panjang dengan lebar atau sebaliknya. Dengan demikian, ada kaitan erat antara ukuran setiap departemen dengan jarak antar departemen, karena ukuran departemen juga akan menentukan struktur tata letak fasilitas.

2.1.1 Metode-metode Pengukuran Jarak Antar Departemen

Tujuan dari mempartisi fasilitas menjadi sub wilayah departemen sehingga dapat meminimalkan biaya pergerakan material total. Tujuan ini berdasarkan pada prinsip penanganan material, bahwa biaya penanganan material bertambah sesuai dengan jarak perjalanan. Jarak tempuh diukur dengan beberapa cara, antara lain *Inpur Output points distance* dan *centroid to centroid distance.*

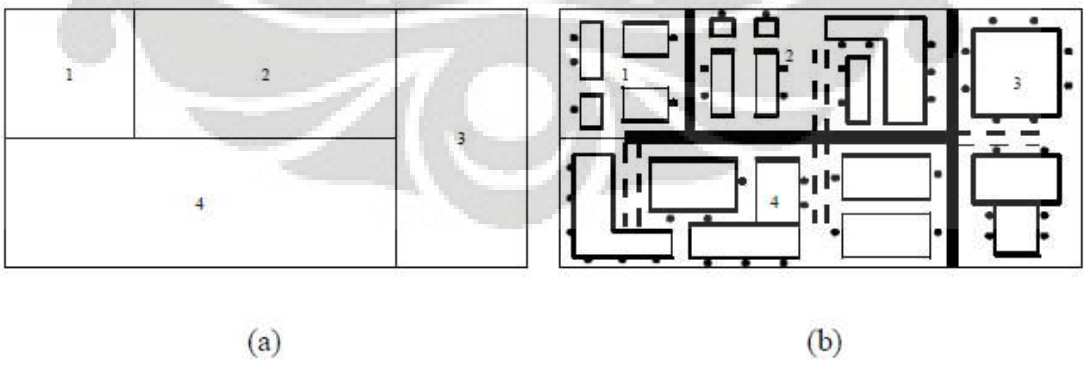
Input-output (I/O) *points distance*: jarak diukur antara titik-titik input output yang spesifik dari dua departemen dan pada beberapa kasus, diukur pula bersama dengan gang-gang dari fasilitas (Tretheway dan Foote, 1994). Kekurangan utama dari

pengukuran akurat ini adalah lokasi-lokasi dari titik I/O dan atau gang-gang masih tidak diketahui hingga detil tata letak sudah didapatkan.

Centroid to centroid distance (CTC) *distance*: jarak dihitung berdasarkan jarak antara dua departemen. Kekurangan dari jarak CTC adalah bentuk dari departemen bisa jadi sangat panjang dan langsing jika batasan bentuk tidak baik. Hal ini karena algoritma yang berdasarkan pada jarak CTC berusaha untuk mendekatkan titik-titik centroid sedekat mungkin (Tate dan Smith,1974).

Dua metode penghitungan dapat digunakan untuk mengkomputasi dua cara pengukuran jarak di atas.Jarak *rectilinier* adalah yang metode paling umum digunakan. Metode ini menghitung jarak menggunakan jalan paralel terhadap aksis *perpendicular* (ortogonal). Armour dan Buffa (1963), Tate dan Smith (1995), dan Liu dan Meller (2007) adalah peneliti-peneliti yang menggunakan jarak *rectilinier* untuk mengkomputasi jarak antara dua departemen. Metode komputasi yang kedua adalah jarak *Euclidean*. Metode ini mengkomputasi jarak dengan menggunakan garis lurus yang menghubungkan dua titik.

Memecahkan UA-FLP akan menghasilkan sebuah tata letak blok, yang menspesifi-kasi lokasi relatif setiap departemen (**Gambar 2.1 (a)**). Hasilnya dapat dikembangkan untuk mendapatkan sebuah tata letak yang detil, yang menspesifikasi lokasi tepat untuk setiap departemen, struktur gang, lokasi titik-titik *input/ouput,* dan tata letak antara setiap departemen (**Gambar 2.1 (b)**).

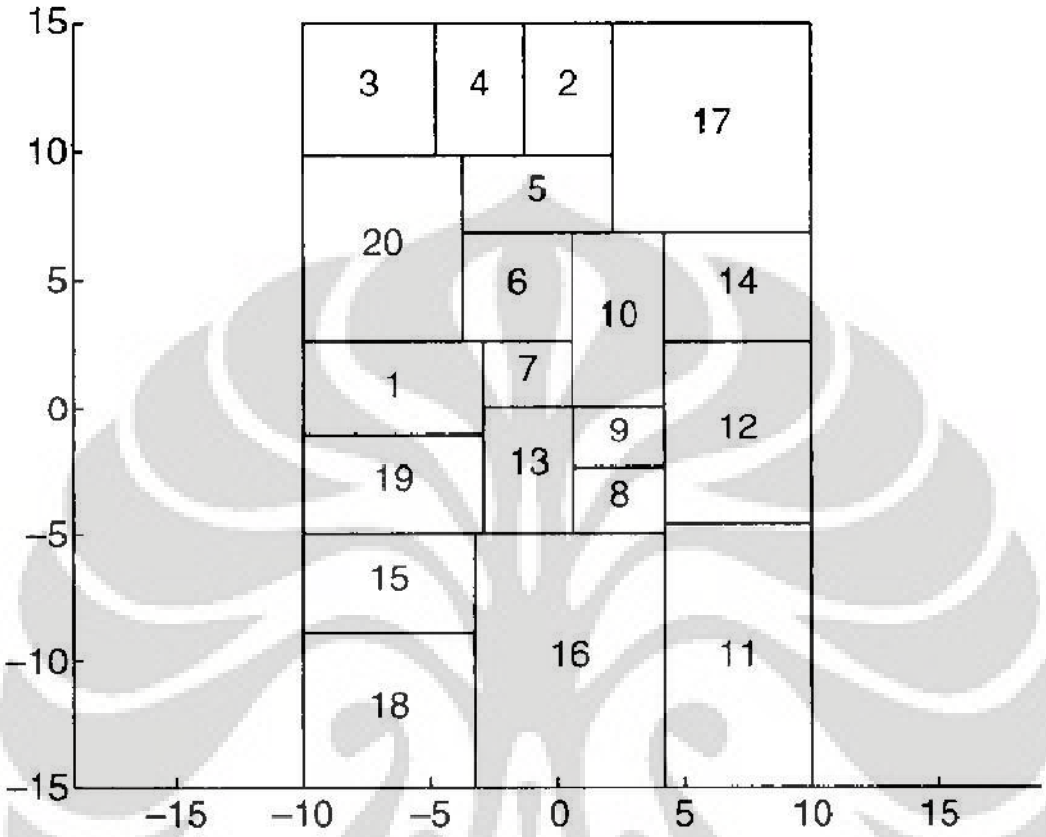


Gambar 2.1 (a) Tata Letak Blok dan (b) Detil Tata Letak

UA-FLP dapat dimodelkan menggunakan berbagai macam representasi. Setiap representasi punya karakteristik dan kompleksitasnya sendiri. Ada empat representasi yang mana dikategorikan berdasarkan pada karakteristik akhir dari tata letak. Mereka adalah model kontinu, model QAP, model *Flexible Bay Structure* (FBS), dan model *Slicing Tree Structure* (STS).

2.1.2 Model Kontinu

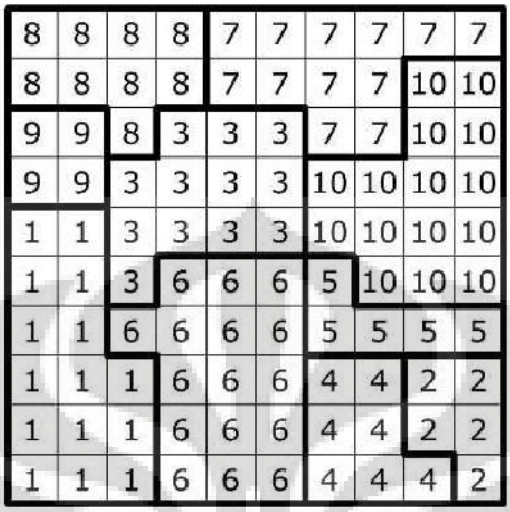
Model kontinu memiliki karakteristik bahwa penempatan setiap departemen dan dimensi pada fasilitas tidak dibatasi atas peraturan apapun selain problem itu sendiri. Hal ini memastikan model memiliki kemungkinan untuk menghasilkan semua solusi tata letak yang mungkin termasuk solusi optimal yang tepat. Meskipun begitu, model ini amat sulit untuk diselesaikan ketika ruang solusinya amat besar. Hal ini mengantar pada pengembangan model lainnya yang lebih mudah untuk dipecahkan. Secara umum, model kontinu sudah digunakan untuk memecahkan UA-FLP dengan *Mixed Integer Linear Programming* (MILP) (Meller dan Gau, 1996) dan program matematis (Anjos dan Vannelli, 2006). **Gambar 2.2** adalah contoh dari tata letak akhir dari model representasi kontinu yang mana tidak dapat dihasilkan dengan representasi-representasi yang lain.



Gambar 2.2 Solusi dari UA-FLP dengan menggunakan representasi kontinu

2.1.3 Quadratic Assignment Model (QAP)

Pada model QAP, seluruh departemen pada problem dicacah menjadi kotak-kotak kecil dengan luas yang sama untuk setiap kotak. Kemudian algoritma akan mencoba mengumpulkan setiap kotak yang memiliki indeks departemen yang sama hingga menjadi sebuah wilayah yang padu. Ini dapat dihasilkan dengan menambahkan arus material buatan yang sangat besar antar kotak dengan indeks departemen yang sama. Hasil dari pemodelan ini dapat dilihat pada **gambar 2.3.**

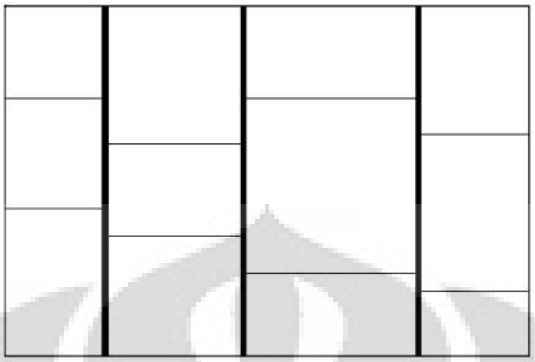


8	8	8	8	7	7	7	7	7	7
8	8	8	8	7	7	7	7	10	10
9	9	8	3	3	3	7	7	10	10
9	9	3	3	3	3	10	10	10	10
1	1	3	3	3	3	10	10	10	10
1	1	3	6	6	6	5	10	10	10
1	1	6	6	6	6	5	5	5	5
1	1	1	6	6	6	4	4	2	2
1	1	1	6	6	6	4	4	2	2
1	1	1	6	6	6	4	4	4	2

Gambar 2.3 Solusi UA-FLP dengan model QAP oleh Hardin dan User (2005)

2.1.4 Flexible Bay Structure Model (FBS)

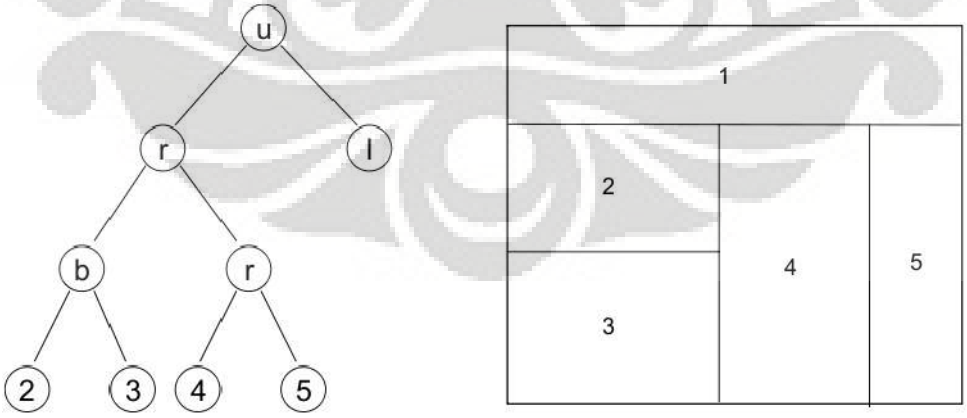
Pada formulasi FBS, penempatan departemen pada UA-FLP menghasilkan kolom-kolom wilayah. Setiap kolom wilayah tidak harus memiliki panjang/lebar yang sama ataupun memiliki jumlah departemen yang sama. Panjang/lebar setiap kolom akan disesuaikan otomatis oleh jumlah departemen yang dikandungnya. Dengan demikian problem menjadi lebih sederhana dan lebih mudah untuk dipecahkan. Kompleksitas problem ditekan menjadi hanya menentukan urutan penempatan departemen-departemen dan jumlah total departemen yang dimiliki setiap kolom wilayah. FBS memiliki keuntungan bahwa kolom wilayah ini akan menjadi kandidat untuk lorong-lorong pada struktur dan ini memfasilitasi pengguna untuk mewujudkan model menjadi desain fasilitas yang nyata (Konak *et al*, 2006).



Gambar 2.4 Solusi untuk UA-FLP dengan model FBS pada Shebanie II (2004)

2.1.5 Slicing Tree Structure Model (STS)

Pada model STS, hasil akhir tata letak UA-FLP dapat dideskripsikan sebagai sebuah representasi pohon. Representasi pohon ini memiliki cabang-cabang yang mewakili departemen-departemen dan karakter "u", "b", "r", dan "l" sebagai cabang-cabang penghubung. Cabang penghubung ini akan berperan sebagai alat pembelah pada formasi wilayah-wilauyah sub-fasilitas. Empat alat pembelah ini adalah "u" untuk *up cut* atau memotong bagian atas, "b" untuk *bottom cut* atau memotong bagian bawah, "r" untuk *right cut* atau memotong bagian kanan, dan "l" untuk *left cut* atau memotong bagian kiri. Cabang-cabang departemen dan alat pembelah mendefinisikan lokasi relatif departemen dan yang berdekatan, begitu pula urutan partisinya. Model STS sudah digunakan oleh Scholz *et al* (2009).



Gambar 2.5 Solusi untuk UA-FLP dengan model STS pada Shebanie II (2004)

2.1.6 Prosedur Heuristik

Sebagian besar dari algoritma UA-FLP dapat diidentifikasi sebagai pembuatan algoritma heuristik ataupun perbaikan algoritma heuristik. Pada kategori sebelumnya, solusi dihasilkan nyaris dari nol. Dapat dianggap bahwa pendekatan heuristik paling sederhana untuk memecahkan UA-FLP, tetapi kualitas dari solusi yang dihasilkan pada umumnya tidak memuaskan. Di sisi lain, perbaikan algoritma heuristik menggunakan iterasi untuk memperbaiki solusi awal. Secara umum, beberapa pendekatan heuristik sudah dipublikasikan pada literatur mencakup CRAFT, SHAPE, NLT, dan MULTIPLE (Meller dan Gau, 1996).

CRAFT (*Computerized Relative Allocation of Facilities Technique*) adalah pendekatan tertua yang dikembangkan oleh Armour dan Buffa (1963). Cara kerja CRAFT adalah dengan memperbaiki hasil awal dengan mengubah-ubah departemen. Pendekatan ini menggunakan dua atau tiga cara untuk mengubah titik-titik pusat dari departemen-departemen yang tidak tetap yang setara dalam luas atau sisi yang bersisian. Untuk setiap perubahan, CRAFT menghitung perkiraan biaya yang ditekan dan kemudian memilih perubahan yang memberikan penekanan biaya terbesar. Hal ini akan dilanjutkan terus hingga tidak dapat dibuat penekanan biaya lagi.

SHAPE dikembangkan oleh Hassan (1986), adalah algoritma konstruksi bahwa pemanfaatan sebuah representasi diskrit dan sebuah fungsi tujuan berbasis pada jarak *rectilinier* antara titik-titik pusat departemen. Pemilihan rangkaian departemen bergantung pada sebuah pemeringkatan, yang mana berbasis pada arus setiap departemen dan nilai arus kritis yang didefinisikan oleh pengguna. Penempatan departemen dimulai dari tengah tata letak. Rangkaian penempatan departemen berikutnya berbasis pada nilai fungsi objektif, dengan departemen-departemen ditempatkan pada setiap empat sisi lainnya. Algoritma ini mudah untuk diimplementasikan; akan tetapi, karena bentuk setiap departemen dikontrol oleh fungsi objektif, bentuk tata letak bisa jadi makin buruk hingga akhir.

NLT (*Nonlinier Optimization Layout Technique*) adalah algoritma konstruksi yang dikembangkan oleh van Camp (1991). Berbasis pada teknik pemrograman non-linier dan memanfaatkan jarak *Euclidean* untuk menghitung jarak antar titik pusat setiap departemen. Pada NLT, ada tiga batasan, departemen tidak dapat bertumpukan, tidak

dapat ditempatkan melewati batas fasilitas, dan tidak dapat ditempatkan di luar area fasilitas. Model yang dibatasi ini kemudian diubah menjadi bentuk yang tidak dibatasi dengan metode pinalti kuadratik titik-titik eksterior. Dengan pendekatan tiga babak, sebuah problem sulit dapat dipecahkan menggunakan solusi dari babak sebelumnya (seperti titik solusi awal). Bentuk dari departemen-departemen ini adalah *rectangular*.

2.2 Algoritma *Differential Evolution*

Beberapa metode metaheuristik seperti SA, GA, *Tabu Search* (TS), dan *Particle Swarm Optimization* (PSO) sudah digunakan untuk memperkirakan solusi untuk UA-FLP yang amat besar. Algoritma SA berasal dari teori mekanik statistik dan berbasis pada analogi antara proses *annealing* dari logam padat dan pemecahan problem optimasi. Secara iteratif GA mencari nilai optimum global dengan menghasilkan solusi-solusi baru melalui proses-proses sejenis melalui pendekatan reproduksi genetis. PSO menggunakan beberapa partikel untuk mewakili solusi-solusi dan membolehkan mereka untuk bergerak pada ruang solusi dengan demikian mereka dapat bekerja sama untuk menghasilkan solusi optimum.

Hingga tulisan ini dibuat, pengembangan metode kontinu untuk memecahkan problem UA-FLP masih sebatas pada penggunaan MILP saja. Oleh karena itulah penulis ingin meneliti implementasi dari algoritma DE sebagai representasi kontinu untuk memecahkan UA-FLP.

Algoritma *Differential Evolution* (DE) adalah sebuah pendekatan untuk meminimalkan fungsi ruang kontinu yang bersifat non-linier dan tidak dapat didiferensiasi (Storn dan Price, 1996). Dengan maksud dari sebuah wilayah pengecekan, pendekatan ini menunjukkan bahwa metode baru bertemu lebih cepat dan lebih pasti dibanding metode-metode optimasi global sebelumnya. Metode baru ini membutuhkan variabel pengontrol lebih sedikit, lebih baik, mudah digunakan, dan fleksibel terhadap komputasi paralel.

DE pertama kali dijelaskan oleh Price dan Storn di ICSI *technical report* pada tahun 1995. Satu tahun kemudian, DE sukses didemonstrasikan di *First International Contest on Evolutionary Optimization* yang diadakan bersamaan dengan *International Conference on Evolutionary Computation* yang diadakan oleh IEEE (*Institute of*

Electrical and Electronics Engineers) dan berhasil memenangkan tempat ketiga. Terinspirasi dari hasil tersebut, Price dan Storn menulis sebuah artikel untuk jurnal Dr. Dobbs ("*Differential Evolution: A Simple Evolution Strategy for Fast Optimization*") yang diterbitkan pada April 1997 dan selanjutnya mereka menerbitkan artikel lagi untuk *Jounal of Global Optimization* ("*Differential Evolution: A Simple and Effiecient Heuristic for Global Opptimization over Continuous Space*"). Artikel-artikel ini memperkenalkan DE ke komunitas internasional dan mendemonstrasikan keunggulan yang dimiliki oleh DE dibandingkan dengan metode heuristik yang lainnya. Pada tahun 1999, Price membuat suatu ringkasan yang berisi penjelasan mengenai algoritma DE ini dalam bentuk buku yang berjudul "*New Ideas in Optimization*".

Setiap pengguna algoritma optimasi mengharapkan lima hal dari sebuah algoritma optimasi, yaitu (Storn & Price, 1997):

1) Kemampuan untuk menangani fungsi biaya yang tidak dapat didiferensiasi, tidak linier, dan multi modal.
2) Kemampuan paralelisasi untuk meraih komputasi fungsi biaya yang intensif.
3) Kemudahan dalam penggunaan, sedikit variabel pengontrol untuk pengendalian minimalisasi. Variabel-variabel ini haruslah tidak mudah berubah dan mudah untuk dipilih.
4) Berkarakter mudah memusatkan, seperti: konsisten dalam pemusatan menuju global optimum pada percobaan bebas secara berurutan.

Algoritma DE didesain agar dapat memenuhi semua syarat di atas (Storn dan Price, 1996).

Agar dapat memenuhi syarat nomor 1, DE didesain menerapkan metode pencarian stokastik langsung. Metode pencarian langsung juga memiliki beberapa kelebihan karena mudah diaplikasikan untuk proses minimalisasi eksperimental, yang mana nilai biaya diturunkan dari eksperimen fisik daripada sebuah simulasi komputer.

Syarat nomor 2 penting untuk komputasi optimasi yang menuntut banyak hal, contohnya, satu evaluasi untuk fungsi biaya membutuhkan waktu bermenit-menit hingga jam. Dalam rangka memperoleh hasil yang dapat digunakan dengan waktu pengolahan yang masuk akal, satu-satunya pendekatan yang dapat berjalan adalah dengan mengandalkan komputer-komputer paralel atau jaringan dari sekumpulan

komputer. DE memenuhi persyaratan nomor 2 dengan sebuah vektor populasi yang mana pengacakan stokastik dari populasi vektor dapat diselesaikan secara mandiri.

Dalam rangka memuaskan syarat nomor 3, akan lebih bagus jika metode minimalisasi dapat mengorganisir mandiri sehingga hanya perlu sedikit input yang dibutuhkan oleh pengguna. Metode DE memiliki kemampuan pengacakan yang bagus pada setiap pada setiap vektor populasi, sehingga pencarian tidak terjebak pada nilai optimum lokal.

Poin terakhir yang penting juga, yang didukung oleh syarat kedua, yakni memiliki kemampuan pemusatan yang baik, sudah menjadi keharusan untuk sebuah algoritma optimasi. Meskipun banyak pendekatan ada untuk mendeskripsikan karakteristik pemusatan dari sebuah metode global optimasi, hanya pengecekan tambahan di bawah berbagai kondisi dapat menunjukkan apakah sebuah metode optimasi dapat memenuhi harapan.

Penjabaran algoritma *Differential Evolution* ini adalah sebagai berikut:

1) Inisialisasi
2) Mutasi
3) Pindah silang *atau Crossover*
4) Evaluasi

2.2.1 Tahap Inisialisasi

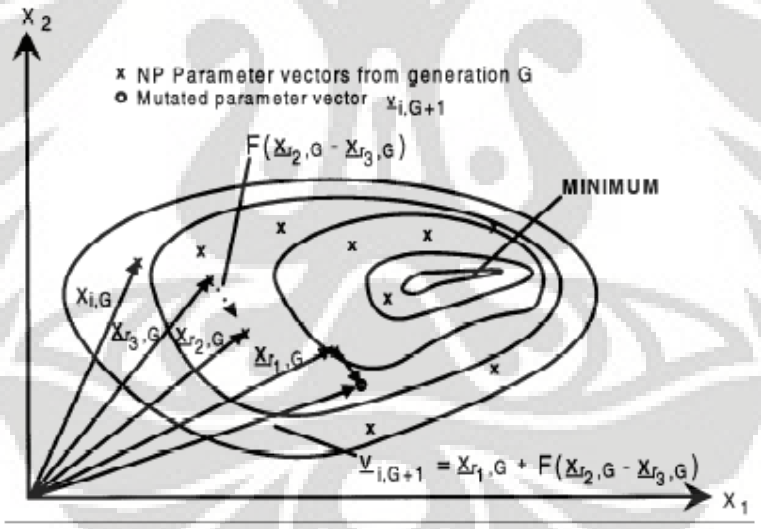
Algoritma *Differential Evolution* (DE) adalah metode pencarian paralel langsung yang memanfaatkan sejumlah NP (*number of population* atau jumlah populasi) vektor dengan parameter dimensi D sebagai populasi dari setiap generasi G. Jumlah populasi konstan selama proses optimasi berlangsung. Jumlah dari NP ini dipilih secara acak dan haruslah cukup besar untuk mengeksplorasi seluruh wilayah pencarian. Sebagai sebuah aturan, kita asumsikan populasi vektor awal dipilih secara acak, selama aturan lain tidak disebutkan. Pengacakan nilai ini dapat disesuaikan dengan skala wilayah pencarian dan dapat disesuaikan dengan kebutuhan pengguna, sehingga setiap nilai yang dihasilkan memiliki bobot tertentu.

2.2.2 Tahap Mutasi

Algoritma DE menghasilkan vektor-vektor parameter yang baru dengan menambahkan vektor differensial yang terbobot, antara dua populasi vektor ke vektor yang ketiga. Untuk setiap vektor target $X_{i,G+1}$ = 1, 2, 3, …, NP, sebuah vektor mutan dihasillkan sesuai dengan rumus berikut:

$$v_{i,G+1} = x_{r_1,G} + F \cdot (x_{r_2,G} - x_{r_3,G}) \tag{2.2}$$

Dengan indeks acak r1, r2, r3 ϵ {1, 2, …, NP}, integer, dengan nilai yang berbeda satu sama lain, dan F > 0. Nilai integer r1, r2, dan r3 yang dipilih juga berbeda dengan indeks target i, sehingga nilai NP lebih besar atau sama dengan empat agar kondisi ini dapat terjadi. F adalah bilangan riil dan faktor konstan ϵ [0, 2] yang mana mengontrol amplifikasi dari variasi differensial ($X_{r2,G} - X_{r3,G}$). **Gambar 2.3** menunjukkan sebuah contoh dua dimensi yang mengilustrasikan vektor-vektor differensial yang ambil bagian dalam generasi $V_{i,G+1}$.



Gambar 2.6 Contoh dari fungsi biaya berdimensi dua, menggambarkan garis kontur dan proses untuk menghasilkan $V_{i,G+1}$ (Storn & Price, 1997)

2.2.3 Tahap Pindah Silang/*Crossover*

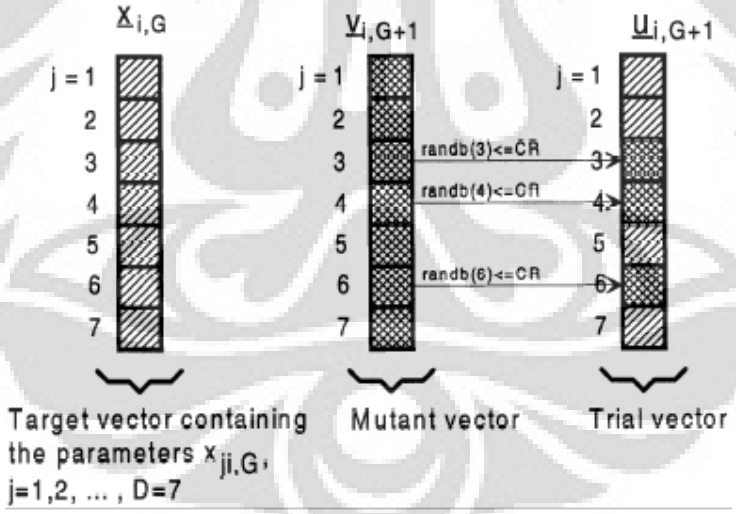
Tahap pindah silang ini adalah tahap yang bertujuan untuk meningkatkan diversitas atau keragaman pengacakan vektor parameter. Untuk menghasilkan vektor parameter hasil dari tahap pindah silang ini, digunakan rumus sebagai berikut:

$$u_{i,G+1} = (u_{1i,G+1}, u_{2i,G+1}, \ldots, u_{Di,G+1}) \tag{2.3}$$

Yang mana:

$$u_{ji,G+1} = \begin{cases} v_{ji,G+1} & \text{if } (randb(j) \leq CR) \text{ or } j = rnbr(i) \\ x_{ji,G} & \text{if } (randb(j) > CR) \text{ and } j \neq rnbr(i \end{cases}$$
$$j = 1, 2, \ldots, D. \tag{2.4}$$

Pada rumus (2.4), *randb*(*j*) adalah evaluasi ke-j dari hasil angka acak yang seragam dengan keluaran ϵ [0, 1]. CR adalah konstanta ϵ [0, 1] yang mana ditentukan oleh pengguna. *rnbr*(*i*) adalah indeks yang dipilih acak dari ϵ 1, 2, …, D yang mana memastikan bahwa $V_{i,G+1}$ mendapatkan paling tidak satu parameter dari $V_{i,G+1}$. **Gambar 2.4** memberikan contoh mekanisme pindah silang untuk vektor 7 dimensi.



Gambar 2.7 Ilustrasi dari proses pindah silang untuk D = 7 parameter (Storn & Price, 1997)

2.2.4 Tahap Seleksi

Untuk memilih apakah vektor percobaan akan menjadi bagian dari generasi G+1 atau tidak, vektor percobaan $U_{i,G+1}$ akan diperbandingkan dengan vektor target $X_{i,G}$ dengan menggunakan kriteria tertentu. Jika vektor $U_{i,G+1}$ menghasilkan nilai dari fungsi biaya lebih rendah dari $X_{i,G}$, maka nilai $X_{i,G}$ sama dengan $U_{i,G+1}$; dengan kata lain, nilai lama dari $X_{i,G}$ akan dibuang.

Pada tahap ini, vektor-vektor yang dihasilkan dari proses-proses sebelumnya akan dikalkulasi nilainya sesuai dengan fungsi objektif yang menjadi permasalahan utama. Dari fungsi biaya inilah akan dipilih vektor yang terbaik dari setiap generasi populasi untuk disimpan sementara dan kemudian diperbandingkan dengan vektor terbaik dari generasi populasi lainnya. Vektor yang terbaik sementara akan dibuang jika vektor terbaik dari generasi populasi berikutnya lebih baik.

Algoritma DE sering disebut-sebut sebagai metode generik, karena pada umumnya, yang perlu diubah-ubah oleh setiap penggunanya hanyalah fungsi evaluasi ini (ditambah dengan proses inisialisasi data, seandainya vektor yang hendak dihasilkan untuk populasi awal dibatasi wilayah pencariannya).

2.2.5 Penentuan Parameter F dan CR

Pada pemakaian algoritma DE, parameter F dan CR, terutama parameter F sangat penting. Nilai F yang terlalu besar akan menyebabkan nilai-nilai optimal terlewat oleh algoritma, sedangkan jika nilai F terlalu kecil, maka algoritma cenderung terjebak pada satu area *local optima* (Karaboğa dan Ökdem, 2004). Penentuan nilai F ini akan menentukan nilai optimal yang dapat dicapai amat bergantung pada problem yang dicari.

Hasil penelitian Karaboğa dan Ökdem (2004) menunjukkan bahwa nilai F untuk lima percobaan yang dilakukannya penting dalam mencapai nilai optimal. Hasil dari penelitiannya dapat dilihat pada **Tabel 2.2** berikut ini. Tabel tersebut menunjukkan generasi yang dibutuhkan untuk mencapai nilai optimal untuk lima problem yang berbeda-beda, baik luas pencarian, karakter titik optimal, maupun kontur dari wilayah pencarian.

Tabel 2.2 Tabel rata-rata angka generasi untuk nilai F yang bervariasi (Karaboğa dan Ökdem, 2004)

F	F1(NP=10, K=0.5, CR=0.8)	F2(NP=10, K=0.5, CR=0.8)	F3(NP=20, K=0.5, CR=0.8)	F4(NP=20, K=0.5, CR=0.8)	F5(NP=10, K=0.5, CR=0.8)
0.2	240	10000^+	165	1100	10000^+
0.4	180	10000^+	145	1150	10000^+
0.6	200	10000^+	140	1750	10000^+
0.8	265	830	130	6700	10000^+
1	360	775	115	10000^+	775
1.5	700	1200	100	10000^+	1520
2	1190	1710	96	10000^+	3160

Hal ini menunjukkan pentingnya penentuan nilai dari F, yang mempengaruhi kemampuan pencarian dari algoritma DE ini.

2.2.6 Implementasi Algoritma DE pada UA-FLP

Berdasarkan dengan fungsi dari algoritma DE yang bekerja pada ruang kontinu (Storn, R. & Price, K. , 21997), algoritma DE sangat cocok diimplementasikan sebagai metode representasi kontinu. Algoritma ini akan menghasilkan nilai variabel-variabel dengan tipe bilangan riil, yang kemudian akan diolah lebih lanjut oleh rumus-rumus fungsi biaya.

Pada implementasi algoritma DE ini, semua kombinasi yang mungkin terjadi pada ruang yang telah disediakan (luas fasilitas) dipertimbangkan dan dihitung sebagai salah satu solusi yang mungkin. Meskipun akan dikenakan sebuah nilai penalti atas kejadian-kejadian di mana departemen-departemen tersebut beririsan.

Tidak semua solusi yang dihasilkan oleh algoritma DE dapat dianggap layak. Yang dimaksud dengan kelayakan di sini adalah jika suatu solusi memiliki departemen-departemen yang saling beririsan. Meskipun secara teoritis hal ini boleh terjadi, namun karena kenyataannya hal tersebut tidaklah riil, maka solusi-solusi yang demikian akan dianggap tidak layak dan tidak akan diproses lebih lanjut.

BAB 3
PENGUMPULAN DATA

Bab ini akan membahas mengenai data-data yang akan digunakan untuk perbandingan dengan metode-metode lainnya. Bab ini tidak membahas mengenai pengambilan data, maupun sumber data tersebut, karena data yang digunakan berasal dari berbagai literatur yang juga membahas UA-FLP, kendati memakai metode dan algoritma yang berbeda.

Tabel 3.1 Daftar Set Problem, Detail Problem, dan Referensinya (Komarudin, 2009)

No	Problem set	Number of departments	Facility size		Common shape constraint	Reference
			Width	Height		
1	O7	7	8.54	13.00	$\alpha_{max} = 4$	Meller *et al.* (1998)
2	FO7	7	8.54	13.00	$\alpha_{max} = 5$	Meller *et al.* (1998)
3	FO8	8	11.31	13.00	$\alpha_{max} = 5$	Meller *et al.* (1998)
4	O9	9	12.00	13.00	$\alpha_{max} = 5$	Meller *et al.* (1998)
5	V10s	10	25.00	51.00	$l_{min} = 5$	van Camp (1989)
6	V10a	10	25.00	51.00	$\alpha_{max} = 5$	van Camp (1989)
7	M11s	11	6.00	6.00	$l_{min} = 1$	Bozer *et al.* (1994)
8	M11a	11	3.00	2.00	$\alpha_{max} = 5$	Bozer *et al.* (1994)
9	M15s	15	15.00	15.00	$l_{min} = 1$	Bozer *et al.* (1994)
10	M15a	15	15.00	15.00	$\alpha_{max} = 5$	Bozer *et al.* (1994)
11	M25	25	15.00	15.00	$\alpha_{max} = 5$	Bozer *et al.* (1994)
12	NUG12	12	3.00	4.00	$\alpha_{max} = 5$	van Camp (1989)
13	NUG15	15	3.00	5.00	$\alpha_{max} = 5$	van Camp (1989)
14	BA12	12	7.00	9.00	$l_{min} = 1$	van Camp (1989)
15	BA12TS	16	7.00	9.00	$l_{min} = 1$	Tate and Smith (1995)
16	BA14	14	6.00	10.00	$l_{min} = 1$	van Camp (1989)
17	BA14TS	14	6.00	10.00	$l_{min} = 1$	Tate and Smith (1995)
18	AB20	20	2.00	3.00	$\alpha_{max} = 1.75$	Armour and Buffa (1963)
19	SC30	30	12.00	15.00	$\alpha_{max} = 5$	Liu and Meller (2007)
20	SC35	35	15.00	16.00	$\alpha_{max} = 4$	Liu and Meller (2007)

Data ini dikumpulkan dan terangkum dari "An Improved Ant System Algorithm For Unequal Area Facility Layout Problems" (Komarudin, 2009). Data detail untuk arus material dan detail ukuran departemen masing-masing kasus terlampir pada halaman lampiran.

Di samping data untuk proses untuk program tersebut di atas, diperlukan juga data mengenai nilai terbaik yang didapat oleh berbagai metode lainnya sebagai perbandingan atas metode DE yang diteliti oleh tulisan ini.

Tabel 3.2 Tabel Nilai Optimal Yang Sudah Ditemukan Oleh Berbagai Metode Yang sudah Diterapkan Untuk UA-FLP (Komarudin, 2009)

No	Problem Set	Tate and Smith (1995)	Konak *et al.* (2006)	Liu and Meller (2007)	Scholz *et al.* (2009)	AS Best objective function value
1	O7	-	-	131.63	132.00	136.58
2	FO7	-	23.12	20.73	-	23.12
3	FO8	-	22.39	22.31	-	22.39
4	O9	-	241.06	235.95	239.07	241.06
5	V10s	-	22,899.65	19,997.00	19,994.10	22,899.64
6	V10a	-	21,463.07	-	-	21,463.10
7	M11s	-	1,317.79	-	-	1,321.35
8	M11a	-	1,225.00	-	-	1,204.15
9	M15s	-	27,781.95	-	-	23,197.80
10	M15a	-	31,779.09	-	-	27,545.30
11	M25	-	-	-	-	1,496.42
12	NUG12	-	265.60	-	-	262.00
13	NUG15	-	526.75	-	-	536.75
14	BA12	-	8,801.33	8,702.00	8,264.00	8,786.00
15	BA12*	-	8,801.33	8,702.00	8,264.00	8,299.50
16	BA12TS	8,861.00	8,600.33	-	-	8,587.05
17	BA14	-	5,004.55	5,004.00	4,712.33	5,004.55
18	BA14*	-	5,004.55	5,004.00	4,712.33	4,913.22
19	BA14TS	5,080.10	4,927.69	-	-	4,927.69
20	AB20	7,205.40	6,890.82	-	-	5,677.83
21	SC30*	-	-	3,706.83	-	3,679.85
22	SC35*	-	-	3,604.12	-	3,962.72

Data-data ini diperlukan untuk mengetahui sudah seberapa baik metode yang digunakan dibandingkan dengan metode-metode yang lain. Tabel di atas menunjukkan perbandingan lima metode yang sudah pernah digunakan, beserta dengan nilai terbaik yang sudah ditemukan.

BAB 4
PENGOLAHAN DATA DAN ANALISA

4.1 Pengolahan Data

Pengolahan data yang dilakukan pada penelitian ini bertujuan hanya mencari nilai optimal yang dapat dicapai oleh metode ini, bukan nilai optimal rekor yang pernah dicapai oleh berbagai metode lainnya; untuk kemudian nilai optimal yang dapat dicapai oleh metode ini akan diperbandingkan dengan nilai-nilai optimal yang dapat dicapai oleh metode-metode lainnya. Sebagai tambahan, dibuat pula statistik nilai-nilai optimal untuk sepuluh kali pencarian yang bertujuan mengukur konsistensi dan kemampuan ukur dari metode DE ini ketika diaplikasikan pada UA-FLP.

4.1.1 Penyusunan Algoritma

Penelitian yang menggunakan algoritma DE ini menggunakan bantuan bahasa pemrograman Java, ditambah program *Netbeans* yang berfungsi menyediakan tampilan yang nyaman bagi pengguna, untuk penulisan kode program. Bersama dengan bahasa pemrograman Java, digunakan pula bantuan Microsoft Excel sebagai alat bantu perhitungan manual dan penulisan rumus-rumus dasar sebelum diterjemahkan ke dalam bahasa program Java.

Bahasa pemrograman *Java* adalah bahasa pemrograman yang pertama kali dikembangkan oleh James Gosling untuk Sun Microsystems dan dirilis pertama kali tahun 1995 sebagai komponen inti dari *Java Platform*. Aplikasi-aplikasi yang berbasis *Java* dikompilasi ke dalam bentuk *byte code* yang nantinya dapat dijalankan oleh *Java Virtual Machine* (JVM) manapun. JVM ini dapat menerjemahkan file kompilasi tersebut tanpa memperdulikan struktur bahasa komputer yang diaplikasikan oleh sistem komputer tersebut.

Ada lima tujuan utama dalam kreasi bahasa *Java*, yaitu:

a) Haruslah "sederhana, *object-oriented*, dan mudah dipahami".
b) Haruslah "kokoh dan aman".
c) Haruslah "netral secara arsitektur dan *portable*".
d) Haruslah mampu dijalankan dengan "performa yang tinggi".

e) Haruslah “mudah diterjemahkan, terurut, dan dinamis”.

Karakter *portability* amat penting, yang artinya program yang ditulis dalam bahasa *Java* harus dapat dijalankan sama persis pada *operating-system* manapun. Dapat dijalankan pada *operating-system* manapun asalkan komputer tersebut sudah memasang *Java Runtime Environment* (JRE).

Penulisan bahasa pemrograman *Java* secara tradisional dapat menggunakan bantuan *command prompt*, akan tetapi cara ini sangat sulit dan melelahkan. Untuk itu biasanya digunakan piranti lunak tambahan disamping memasang JRE untuk *Java*, yaitu *integrated development environment* (IDE) untuk mengembangkan program *Java*, ditambah dengan program ruang kerja untuk tampilan yang memudahkan pembuat program. Program ruang kerja yang dipilih oleh penulis kali ini adalah Netbeans versi 1.6.8.

Alasan pemilihan bahasa pemrograman *Java* ini karena keunggulan-keunggulan bahasa pemrograman *Java*, yaitu sebagai berikut:

a) *Object oriented*.

   Maksudnya pemrograman ini berorientasi objek, yang mana setiap proses atau objek dapat memiliki karakter-karakternya yang unik. Dari situ dapat diadakan pengolahan data yang lebih efektif.

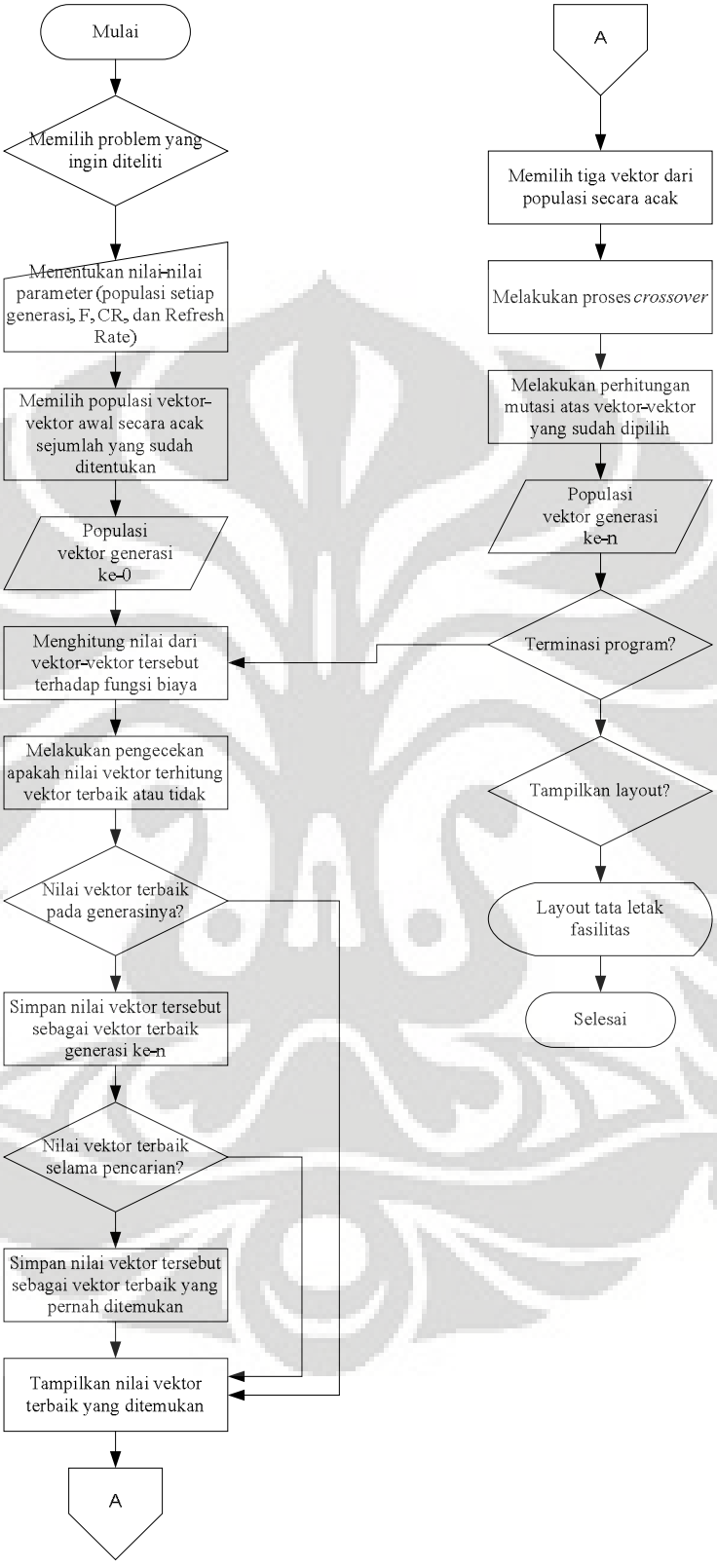
b) Perpustakan *script code*.

   Perpustakaan kode pemrograman ini sudah disediakan, dan fungsinya adalah, semisal ingin membuat sebuah objek ataupun proses, tidak perlu lagi membuat ulang dari awal jika sudah ada pada JRE, cukup impor kode tersebut, kemudian dipanggil sesuai kebutuhan.

c) Efisien.

   Pemrograman dengan *Java* menjadi efisien, karena setiap rumus ataupun metode yang akan digunakan berulang-ulang pada tahapan yang berbeda tidak perlu ditulis ulang berkali-kali, tapi cukup ditulis sebagai *method* tersendiri, kemudian dipanggil jika dibutuhkan.

Langkah-langkah yang diambil dalam penyusunan program ini dapat dilihat pada **Gambar 4.1** berikut ini:



Gambar 4.1 Diagram Alir Optimasi Algoritma DE Pada Program Java

Input data pada penelitian kali ini tidak dilakukan, karena program yang dibuat memang untuk tujuan penelitian, dan terbatas pada dua puluh kasus yang diteliti saja, sehingga masalah pemasukan data pada program tidak perlu fleksibel. Pada program ini bahkan input data hanya bisa dilakukan dengan mengubah data yang sudah ditulis pada *source code*–nya secara langsung. Di luar dari itu, pengguna hanya bisa memilih set data yang hendak diteliti.

Proses penyimpanan data yang dilakukan pada algoritma ini terbagi menjadi dua, yaitu data vektor terbaik generasi dan data vektor terbaik selama pencarian. Yang dimaksud vektor di sini adalah variabel-variabel yang menjadi individu dari sebuah populasi pada suatu generasi. Data-data *output* dari program ini adalah:

a) Nilai optimal untuk problem X, untuk penelitian ke-n.
b) Data vektor yang menghasilkan nilai optimal, yang terdiri dari panjang, lebar, koordinat X, dan koordinat Y untuk semua departemen yang terlibat pada problem tersebut.
c) Gambar tata letak yang dibentuk oleh vektor-vektor yang menghasilkan nilai optimal tersebut. Selanjutnya gambar tata letak terbaik ini disebut desain yang optimal.

Di samping bahasa pemrograman Java, perlu dibuat juga perhitungan yang dibuat di Microsoft Excel. Fungsi dari perhitungan Excel ini adalah:

a) Sebagai dasar proses algoritma perhitungan yang akan ditulis pada bahasa pemrograman Java.
b) Sebagai tempat eksperimen penulisan algoritma yang benar agar tidak terjadi kesalahan alur pada algoritma.
c) Sebagai alat bantu validasi atas program Java yang sudah dibuat.
d) Sebagai tempat perencanaan dasar atas program Java yang akan dibuat. Dengan demikian program Java menerjemahkan Excel ke dalam bahasa pemrograman Java.

Perencanaan yang dilakukan pada Excel sebatas pada perhitungan evaluasi vektor saja, tidak mencakup algoritma DE yang menjadi inti dari

penelitian ini. Hal ini disebabkan Rainer & Storn selaku pengembang algoritma DE sudah membuat dan menerbitkan algoritma DE ini pada beberapa bahasa pemrograman, salah satunya Java sebagai *source code* yang dapat dipergunakan dengan bebas selama tetap mencantumkan lisensi yang sudah dibuat pada setiap program yang menggunakan algoritma buatan mereka.

Berikut ini adalah kutipan dari lisensi tersebut: (Storn dan Price, 1997)

"*This is the kernel routine for the DE optimization.*

Authors: Mikal Keenan
Kenneth Price
Rainer Storn

This program implements some variants of Differential Evolution (DE) as described in part in the techreport tr-95-012.ps of ICSI. You can get this report either via ftp.icsi.berkeley.edu/pub/techreports/1995/tr-95-012.ps.Z or via WWW: http://http.icsi.berkeley.edu/~storn/litera.html A more extended version of tr-95-012.ps has appeared in the Journal of global optimization.

You may use this program for any purpose, give it to any person or change it according to your needs as long as you are referring to Rainer Storn and Ken Price as the originators of the the DE idea."

Pada permasalahan UA-FLP ini, ada tiga variabel independen dan satu variabel dependen yang menjadi objek pencarian oleh algoritma DE. Ketiga variabel independen tersebut adalah panjang departemen, koordinat X, dan koordinat Y. Sedangkan satu variabel dependen ini adalah lebar departemen, yang mana besarnya bergantung pada hasil pembagian luas departemen (ditentukan) dengan panjang departemen (didapat dari algoritma DE).

Agar dapat mengaplikasikan metode DE untuk mendapatkan solusi dari UA-FLP, diperlukan beberapa penyesuaian, berupa penambahan rumus-rumus pembatas atau konstrain, kondisi-kondisi pembatas, serta nilai penalti agar algoritma DE dapat berfungsi dengan baik.

Yang termasuk dalam rumus-rumus pembatas tersebut antara lain:

1) $= \overline{\qquad\qquad}$ (4.1)

$=$ (4.2)

Kedua rumus ini mengatur batas minimal sebuah departemen, dengan nilai yang sudah diketahui luas setiap departemen, dan nilai yang ditentukan untuk *Rasio* dan I_{min}.

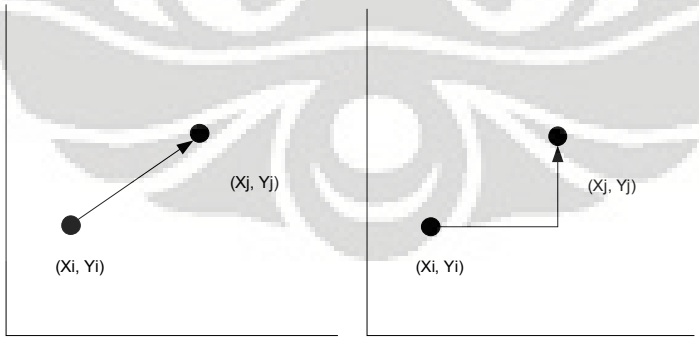
2) $=$. (4.3)

$=$ (4.4)

Kedua rumus ini mengatus batas maksimal sebuah departemen, yang mana besaran nilainya berkaitan erat dengan luas departemen dan panjang lebar minimum departemen tersebut.

3) $= | - | + | - |$ (4.5)

Adalah rumus yang menghitung jarak antara dua departemen, dihitung dari titik pusat atau *centroid* masing-masing departemen. Jarak yang diperhitungkan bukan merupakan jarak cartesian yang menghitung jarak lurus antar dua titik, melainkan jarak rectilinier yang menghitung total jarak koordinat x dan y antara kedua departemen tersebut.



(a) (b)

Gambar 4.2 (a) Jarak menurut metode cartesian, dan (b) jarak menurut metode rectilinier

Alasan mengapa penulis memakai metode penghitungan memakai metode rectilinier adalah sebagai berikut:

1) Agar hasilnya dapat diperbandingkan dengan algoritma ACO
2) Agar desain tata letak yang dihasilkan tidak bertumpukan

Beberapa asumsi juga diambil dalam pencarian nilai fungsi biaya ini. Asumsi-asumsi yang diambil adalah sebagai berikut:

1) Nilai biaya penanganan material diasumsikan satu.
2) Departemen-departemen mungkin dan boleh bertumpukan, selama nilai penalti yang dikenakan atas wilayah yang bertumpukan tersebut memang lebih baik dibanding jika tidak bertumpukan.

Selain rumus-rumus konstrain tersebut di atas, dibutuhkan pula rumus lain yang bertugas mencegah variabel-variabel keluar dari kondisi pembatas. Yang dimaksud dengan kondisi pembatas sendiri adalah berbagai macam syarat yang membatasi besaran dari vektor-vektor variabel yang hendak dicari. Contoh dari kondisi pembatas ini: jika sebuah nilai yang didapat dari algoritma DE untuk variabel panjang departemen setelah proses mutasi lebih besar dari panjang maksimum atau lebih kecil dari panjang minimum, maka nilai fungsi biaya untuk vektor ini akan dikenai penalti yang amat besar, sehingga tidak mungkin vektor ini terpilih sebagai vektor terbaik.

Rumus-rumus evaluasi yang dimaksud di sini bertugas menerjemahkan vektor yang dihasilkan oleh algoritma menjadi variabel yang dapat digunakan oleh fungsi biaya. Rumus-rumus tersebut antara lain:

Jika vektor yang dihasilkan oleh algoritma adalah $\begin{matrix}1\\2\\3\end{matrix}$, maka

$$Panjang\ Departemen = MinPL + a1 \times (MaxPL - MinPL) \tag{4.6}$$

dan

$$Lebar\ Departemen = \frac{Luas\ Departemen}{Panjang\ Departemen} \tag{4.7}$$

Pada kedua rumus ini, nilai vektor yang dipakai hanyalah a1, yang mana a1 adalah vektor untuk besaran panjang. Rumus (4.6) sesungguhnya adalah rumus jangkauan minimal dan maksimal panjang yang diperbolehkan, karena ada konstrain rasio departemen. Lebar departemen dalam hal ini merupakan variabel dependen terhadap panjang departemen, sehingga tidak perlu diwakilkan oleh vektor lagi.

Jika pada bab 2 sudah dijelaskan mengenai rumus dasar untuk menghitung biaya total, maka rumus dasar tersebut perlu ditambahkan dengan rumus penalti untuk mendukung aplikasi dari algoritma DE ini, agar departemen-departemen tidak menjadi bertumpukan, karena algoritma DE hanya melihat jarak pada rumus dasar tersebut, sehingga jarak tersebutlah yang akan diminimalisasi menjadi 0. Berikut ini adalah rumus biaya total yang dimodifikasi:

$$cost_{ij} = distance_{ij} \times material\ flow_{ij} + intersection\ area_{ij} \times \frac{(total\ length + total\ width) \times totalflow \times penalty\ modifier}{(facility\ length \times facility\ width)} \quad (4.8)$$

dengan,

intersection $area_{ij}$ = luas irisan antara departemen i dengan departemen j

total length = total panjang semua departemen

total width = total lebar semua departemen

total flow = total nilai arus lalu lintas material

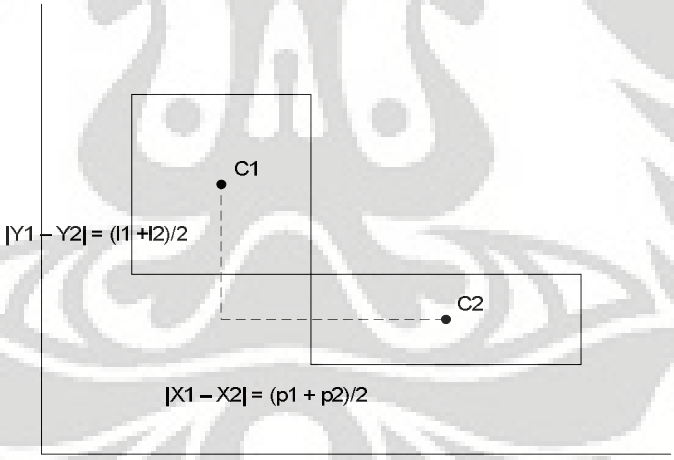
penalty modifier = besarnya pengali penalti yang dapat diubah-ubah sesuai keperluan. Biasanya nilai yang ditetapkan adalah 5 atau 10.

Disamping penalti yang dikenakan pada luas area irisan yang terjadi, penulis juga menetapkan nilai-nilai penalti apabila nilai-nilai vektor $\begin{matrix}1\\2\\3\end{matrix}$ keluar dari jangkauan nilai yang sudah ditetapkan. Yang dimaksud jangkauan nilai tersebut adalah besaran minimal dan maksimal yang boleh dan riil. Sebagai contoh: nilai a1 sebagai indeks vektor panjang tidak boleh minus, karena jika nilainya minus, maka besaran panjang bisa dibawah batas minimum atau lebih kecil dari MinPL, bahkan nilainya bisa minus. Penalti-penalti tersebut sudah ditetapkan setinggi-tingginya, dalan hal ini nilainya ditetapkan sebesar sepuluh

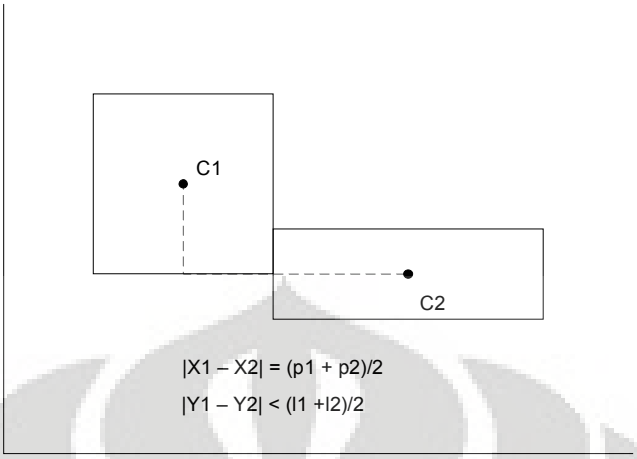
juta, dengan tujuan vektor-vektor yang tidak layak tersebut tidak dipertahankan oleh algoritma sebagai solusi. Penalti-penalti tersebut dikenakan pada individu bila:

- Setiap kali ada bagian dari salah satu departemen yang keluar dari area fasilitas.
- Setiap kali ada vektor a1 yang lebih kecil dari 0 atau lebih besar dari 1.

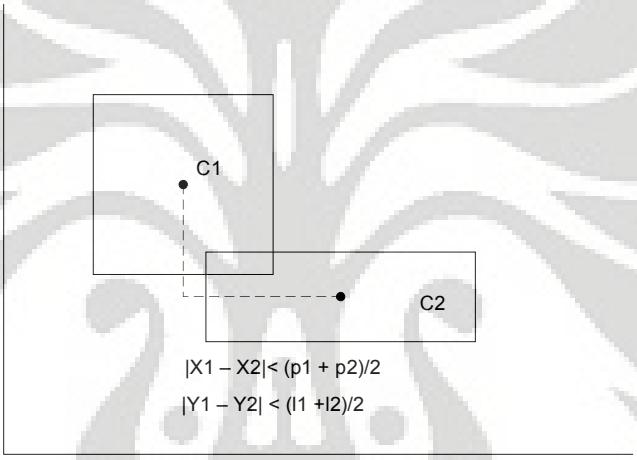
Untuk menghitung wilayah area irisan antara dua departemen, maka diperlukan algoritma khusus untuk menghitungnya. Logika hitung dari algoritma ini sederhana, yaitu jika jarak kedua titik pusat dari persegi lebih kecil dari total setengah panjang (sumbu X) kedua departemen, maka kedua persegi panjang beririsan pada sumbu X. Demikian pula halnya dengan sumbu Y. Jika kedua persegi panjang beririsan pada sumbu X dan sumbu Y, maka kedua departemen tersebut beririsan, dengan luas irisan sebesar panjang irisan pada sumbu X dikalikan dengan panjang irisan pada sumbu Y. Untuk lebih jelas, dapat dilihat pada ilustrasi **Gambar 4.3**.



(a)



(b)



(c)

Gambar 4.3 (a) Kondisi tidak beririsan pada sumbu X dan Y, (b) Kondisi beririsan pada sumbu Y, (c) Kondisi beririsan pada sumbu X dan Y

Ditambahkan pula algoritma tambahan berikut ini untuk menjamin bahwa nilai dari panjang dan lebar area irisan tidak lebih besar dari panjang dan lebar salah satu departemen yang berkaitan. Jika kode di bawah ini tidak ditambahkan, seringkali terjadi perhitungan luas irisan lebih besar dari salah satu persegi panjang, yang mana hal tersebut tidaklah mungkin terjadi.

```
{
    if(X_intersection > X1)
      X_intersection = X1;
    if(X_intersection > X2)
```

```
            X_intersection = X2;
    }

    {
        if(Y_intersection > Y1)
            Y_intersection = Y1;
        if(Y_intersection > Y2)
            Y_intersection = Y2;
    }
```

Di samping perbaikan yang dilakukan atas rumus evaluasi, turut ditambahkan pula alat bantu untuk menggambarkan desain optimal yang ditemukan oleh algoritma DE. Selain menampilkan desain optimal, perintah ini juga akan menampilkan data panjang, lebar, dan posisi (koordinat xy) setiap departemen. Perintah tersebut aktif bila:
Start → pause → resume → layout + data → stop

4.1.2 Verifikasi Program

Verifikasi program adalah proses pemeriksaan program yang sudah dibuat, apakah algoritma dan rumus perhitungan yang dibuat sudah sesuai dengan yang direncanakan sejak awal (sesuai dengan logika diagram alur). Dengan demikian nilai keluaran dari program ini juga dapat dipastikan adalah hasil keluaran yang sesuai dengan perencanaan kita. Beberapa proses verifikasi yang dilakukan dalam proses pembuatan program ini adalah sebagai berikut:

a) Memastikan kebenaran logika berpikir, yaitu kesesuaian penulisan *source code* dengan konsep algoritma *Differential Evolution* dan dapat dijalankan dengan baik.
b) Memastikan bahwa program sudah memasukkan semua batasan masalah yang ada.
c) Mengecek fungsi tujuan.
d) Memastikan semua data terhitung dan terproses.

Khusus untuk algoritma DE tidak dilakukan verifikasi lebih lanjut, karena penulis sudah mendapatkan *source code* dari pembuat algoritma DE tersebut, yakni Rainer Storn dan Kenneth Price. Verifikasi dilakukan sebatas pada rumus fungsi tujuan saja, yang terdapat pada setiap kelas problem DE.

Verifikasi program dilakukan dengan mengacu pada diagram alir pada **Gambar 1.4**. Verifikasi yang dilakukan sudah menunjukkan bahwa program Java yang dibuat berdasar pada Excel sudah mengikuti diagram alir tersebut.

4.1.3 Validasi Program

Validasi program adalah proses untuk memastikan program menghasilkan nilai keluaran yang benar. Artinya hasil perhitungan yang dilakukan oleh program bernilai sama dengan perhitungan manual. Validasi dapat dilakukan dengan data *dummy* atau fiksi, akan tetapi pada validasi kali ini, angka yang digunakan adalah data-data hasil dari eksperimen yang sesungguhnya. **Tabel 4.1** adalah contoh data vektor detail yang memberikan gambaran vektor desain optimal. **Tabel 4.1** adalah hasil dari program yang menjadi bahan validasi program Java yang sudah dibuat.

Dari **Tabel 4.2** dapat kita lihat bahwa nilai biaya total untuk percobaan ini adalah 161,82. Hasil yang didapat dari perhitungan manual ini sama dengan nilai yang didapat dari perhitungan program Java yang dibuat. Dengan demikian dapat kita ambil kesimpulan bahwa program yang sudah dibuat tidak perlu diragukan lagi validitasnya.

Tabel 4.1 Daftar Input Data

No	Departemen	Luas	Ukuran		Koordinat Centroid	
			P	L	x	y
1	nomor 1	238	3.85	4.16	1.92	2.08
2	nomor 2	112	3.85	4.16	1.92	8.58
3	nomor 3	160	4.69	3.41	6.19	1.70
4	nomor 4	80	4.69	7.67	6.19	7.25
5	nomor 5	120	4.69	1.92	6.19	12.04
6	nomor 6	80	3.85	2.34	1.92	5.33
7	nomor 7	60	3.85	2.34	1.92	11.83
	Total	850	29.47	25.99		

4.2 Aturan Dalam Menjalankan Program

Aturan dalam menjalankan program adalah prosedur-prosedur yang bertujuan untuk menyeragamkan proses menjalankan program, sehingga nilai keluaran yang diperoleh dari setiap percobaan memiliki kualitas yang sama. Kualitas setiap percobaan sedapat mungkin sama, agar dapat diperbandingkan dengan baik.

Kualitas yang dimaksud disini tidak dilihat dari nilai biaya total yang dihasilkan, melainkan dari gambar tata letak yang keluar. Kualitas sebuah gambar tata letak dapat dianggap baik jika:

a) Luas irisan antar departemen minimal (sebisa mungkin tidak ada),
b) Kerenggangan antar departemen minimal (sebisa mungkin tidak ada).

Aturan-aturan tersebut antara lain:

- Melakukan tes sebanyak dua kali untuk setiap percobaan untuk memperkirakan generasi yang diperlukan agar desain tata letak menjadi rapi. Jumlah generasi dari perkiraan tersebut menjadi acuan generasi minimal yang harus dijalani pada percobaan tersebut. Tes ini juga diperlukan untuk menentukan skala dari gambar, nilai penalti modifikasi, dan nilai dari F dan CR yang tepat untuk percobaan tersebut.
- Tidak menghentikan program selama masih terjadi perbaikan nilai biaya total yang signifikan (minimal terjadi penurunan nilai sebesar 1 poin).
- Mengulang percobaan jika nilai total biaya macet pada nilai 10 kali lipat lebih besar dari nilai terbaik yang diketahui dari literatur.

Aturan-aturan yang dibuat disini tidak bersumber dari literatur manapun, karena: jumlah generasi, nilai F, nilai CR, dan waktu yang dibutuhkan sebuah problem untuk mencapai nilai optimalnya sangat bervariasi, tergantung pada kompleksitas dari problem tersebut. Dengan demikian hal yang perlu diperhatikan dari pembuatan aturan ini adalah memastikan nilai optimal untuk desain tata letak tersebut sudah tercapai.

4.3 Hasil

Hasil penelitian ini adalah data biaya total optimal untuk sepuluh kali menjalankan program perhitungan untuk setiap kasus dari total dua puluh kasus yang didapat dari berbagai literatur. Dengan demikian totalnya ada dua ratus percobaan yang dilakukan. Di samping data rekaman tersebut, dicatat pula nilai terbaik dan terburuk dari kesepuluh kali pemrosesan data tersebut. Penjabaran detail hasil penelitian dapat dilihat pada **Tabel 4.3**

Penelitian juga mencatat data statistik nilai terbaik, terburuk, dan rata-rata untuk setiap problem. Nilai terbaik dicatat dengan tujuan mengetahui rekor nilai optimal untuk dibandingkan dengan nilai terbaik yang dapat dicatat oleh metode lain. Nilai terburuk dan nilai rata-rata dicatat untuk kemudian diperbandingkan dengan nilai terbaik yang dapat dicapai oleh metode ini, dengan tujuan mengetahui konsistensi dari metode ini, apakah dapat mencapai nilai yang sama walaupun penelitian diulang berkali-kali.

Tabel 4.4 memberikan gambaran yang jelas mengenai perbandingan-perbandingan antar metode yang sudah pernah diterapkan untuk problem UA-FLP ini berdasarkan nilai optimum yang sudah pernah dicapai oleh masing-masing metode. Dari tabel tersebut dapat kita lihat bahwa secara umum, metode DE dengan representasi kontinu ini hanya mampu memberikan peningkatan nilai optimum untuk kasus SC30 saja. Lebih lanjut penulis akan menganalisis satu per satu kinerja metode DE pada setiap kasus yang diteliti berdasar pada uji *Reliability Analysis* dengan menggunakan bantuan program SPSS atas sepuluh percobaan.

4.3.1 Uji Konsistensi Algoritma DE

Uji konsistensi dengan menggunakan *reliability analysis* ini diperlukan untuk melihat kelayakan metode, apakah layak diaplikasikan pada kasus nyata atau tidak. Kesepakatan internasional menyebutkan bahwa nilai dari pengujian ini dianggap layak bila lebih besar sama dengan 0.70. (faculty.chass.ncsu.edu/ garson/PA765/reliab.htm) Algoritma DE akan diuji konsistensinya untuk setiap problem, untuk mengetahui apakah ada hubungan tertentu antara karakteristik problem dengan kinerja dari algoritma ini. Metode pengujian dengan *reliability*

analysis ini dipilih oleh penulis dengan pertimbangan sederhana dan mudah untuk dicari nilai ujinya.

Tabel 4.2 Tabel Hasil Uji Konsistensi Dengan *Reliability Analysis*

Problem Set	Cronbach's Alpha if Item Deleted	Std. Deviation
O7	.354	40.85%
FO7	.355	4.25%
FO8	.355	9.92%
O9	.354	17.58%
V10s	.329	21.36%
V10a	.356	3567.61%
M11s	.383	6276.17%
M11a	.382	2495.35%
M15s	.097	1337.54%
M15a	.096	16049.85%
M25	.382	12447.70%
NUG12	.355	2120.54%
NUG15	.355	109.49%
BA12	.348	511.39%
BA12TS	.290	5470.65%
BA14	.343	332.51%
BA14TS	.409	3588.85%
AB20	.350	919.20%
SC30	.336	600.73%
SC35	.375	1238.64%

Dari tabel tersebut di atas, dapat dilihat bahwa algoritma DE tidak konsisten untuk diterapkan pada kasus mana pun. Dengan demikian algoritma ini masih jauh dari layak untuk diaplikasikan pada kasus nyata. Lebih jauh, hal ini menjadi kekurangan fatal dari algoritma DE, diperlukan pengembangan lebih lanjut agar algoritma ini dapat diterapkan pada kasus nyata

4.3.2 Karakteristik Problem-problem Yang Diteliti

Karakteristik dari problem-problem yang akan dianalisis di sini perlu untuk dibahas karena tidak setiap problem memiliki karakteristik yang sama. Salah satu alasan penting yang menyebabkan perlunya pembahasan tersebut

adalah karena pada beberapa kasus algoritma DE mampu mendapatkan baik nilai optimal yang mendekati nilai rekor, maupun desain optimal yang tidak terkena nilai penalti sedikitpun.

Karakteristik-karakteristik dari problem-problem tersebut adalah sebagai berikut:

a) Memiliki arus material sebesar satu untuk semua arus material. Contoh dari karakteristik ini adalah problem FO7.
b) Memiliki arus material yang bervariasi. Umumnya problem-problem yang diteliti memiliki karakteristik ini.
c) Luas departemen bervariasi. Hampir semua problem memiliki karakteristik ini.
d) Luas semua departemen persis sama. Contoh dari karakteristik ini adalah problem NUG12, yang memiliki luas sebesar satu satuan untuk semua departemennya.
e) Luas total semua departemen persis sama dengan luas fasilitas. Contoh dari karakteristik ini adalah problem O7, O9, dan lain-lain.
f) Luas total semua departemen tidak persis sama dengan luas fasilitas, sehingga akan ada wilayah-wilayah kosong yang tidak terisi oleh departemen. Contoh dari karakteristik ini adalah problem SC30, SC35, dan lain-lain.

4.3.3 Kelayakan Solusi Yang Dihasilkan

Seperti yang sudah dibahas pada bab 2 bahwa tidak semua solusi dapat dianggap layak karena tidak riil, maka dilakukan pendataan solusi-solusi mana saja yang tidak layak. Data-data solusi mana saja yang layak dan tidak layak dapat dilihat pada **Tabel 4.6** (detail dan desain dari solusi yang tidak layak tidak ditampilkan).

Tabel 4.3 Tabel Perhitungan Manual Data O7 Percobaan Ke-1

No	Departemen i	Departemen j	Jarak (Dij)	Pengecekan Area Irisan								Luas area overlapping	Penalty Constanta	Total Material Flow (Fij)
				((X1) + (X2))/2	X (1-2)	Pembanding sumbu X	Panjang area irisan	((Y1) + (Y2))/2	Y (1-2)	Pembanding sumbu Y	Lebar area irisan			
	1	2	6.50	3.85	0.00	3.85	3.85	4.16	6.50	-2.34	0.00	0.00	0.00	0.00
		3	4.65	4.27	4.27	0.00	0.00	3.78	0.38	3.41	3.41	0.00	0.00	0.00
		4	9.43	4.27	4.27	0.00	0.00	5.92	5.16	0.75	0.75	0.00	0.00	47.16
		5	14.23	4.27	4.27	0.00	0.00	3.04	9.96	-6.92	0.00	0.00	0.00	0.00
		6	3.25	3.85	0.00	3.85	3.85	3.25	3.25	0.00	0.00	0.00	0.00	0.00
		7	9.75	3.85	0.00	3.85	3.85	3.25	9.75	-6.50	0.00	0.00	0.00	9.75
	2	3	11.15	4.27	4.27	0.00	0.00	3.78	6.88	-3.09	0.00	0.00	0.00	0.00
		4	5.61	4.27	4.27	0.00	0.00	5.92	5.54	0.37	0.37	0.00	0.00	16.82
		5	7.73	4.27	4.27	0.00	0.00	3.04	4.79	-1.76	0.00	0.00	0.00	0.00
		6	3.25	3.85	0.00	3.85	3.85	3.25	6.71	-3.46	0.00	0.00	0.00	0.00
		7	3.25	3.85	0.00	3.85	3.85	3.25	6.50	-3.25	0.00	0.00	0.00	3.25
	3	4	5.54	4.69	0.00	4.69	4.69	5.54	5.54	0.00	0.00	0.00	0.00	11.08
		5	10.34	4.69	0.00	4.69	4.69	2.66	10.34	-7.67	0.00	0.00	0.00	0.00
		6	7.90	4.27	4.27	0.00	0.00	2.87	3.63	-0.75	0.00	0.00	0.00	0.00
		7	14.40	4.27	4.27	0.00	0.00	2.87	10.13	-7.25	0.00	0.00	0.00	14.40
	4	5	4.79	4.69	0.00	4.69	4.69	4.79	4.79	0.00	0.00	0.00	0.00	19.18
		6	6.18	4.27	4.27	0.00	0.00	5.01	1.91	3.09	3.09	0.00	0.00	24.73
		7	8.85	4.27	4.27	0.00	0.00	5.01	4.58	0.42	0.42	0.00	0.00	0.00
	5	6	10.98	4.27	4.27	0.00	0.00	2.13	6.71	-4.58	0.00	0.00	0.00	0.00
		7	4.48	4.27	4.27	0.00	0.00	2.13	0.21	1.92	1.92	0.00	0.00	8.96
	6	7	6.50	3.85	0.00	3.85	3.85	2.34	6.50	-4.16	0.00	0.00	0.00	6.50
												Total Material Flow		161.82
												Total Penalty		0.00
												Total Cost		161.82

Tabel 4.4 Sepuluh Data Hasil Pencarian Problem O7

No	Problem Name	Department	F	CR	Generation	AS Best	AS Worst	AS Average	1	2	3	4	5	6	7	8	9	10
1	O7	7.00	0.80	0.80	40,000	151.10	292.68	185.00	161.82	178.92	292.68	165.33	172.15	162.65	181.42	208.43	175.50	151.10

Tabel 4.5 Tabel Hasil Penelitian

No	Problem Name	DE Best	DE Worst	1	2	3	4	5	6	7	8	9	10
1	O7	151.10	208.43	161.82	178.92	292.68	165.33	172.15	162.65	181.42	208.43	175.50	151.10
2	FO7	27.42	41.23	39.92	33.56	40.28	39.67	35.05	36.36	33.52	27.42	41.23	37.86
3	FO8	34.81	67.98	54.98	41.52	60.82	34.81	44.88	67.98	49.47	45.67	42.75	43.85
4	O9	292.23	347.62	292.23	336.63	318.92	316.79	306.58	317.29	326.50	347.62	326.69	293.28
5	V10s	34,809.68	41,491.40	34,809.68	41,491.40	33,689.88	37,259.50	35,594.83	44,671.98	40,512.37	36,766.80	40,494.85	35,048.44
6	V10a	28,367.79	48,886.85	38,194.76	48,886.85	42,303.19	44,957.42	32,396.90	40,511.50	32,180.07	40,244.32	36,645.24	28,367.79
7	M11s	0.00	0.00	4,535.84	5,284.32	6,704.01	9,200.86	2,581.27	8,987.32	1,679.34	6,305.24	7,830.60	5,806.25
8	M11a	1,727.78	3,255.72	4,538.84	4,469.09	3,255.72	2,464.95	2,644.04	1,727.78	2,443.86	6,061.13	2,582.61	2,439.41
9	M15s	60,621.17	77,405.90	63,962.01	91,397.61	77,405.90	106,276.66	87,257.91	79,925.36	109,624.39	60,621.17	74,451.57	85,462.54
10	M15a	0.00	0.00	64,944.00	95,232.84	87,385.14	76,810.59	71,632.34	61,768.20	92,254.70	64,881.49	67,592.53	85,954.97
11	M25	0.00	0.00	9,010.82	10,712.42	8,896.91	5,365.65	11,186.42	6,091.44	7,376.55	10,333.47	8,420.02	5,660.44
12	NUG12	420.38	443.26	470.83	414.54	426.81	443.06	443.26	479.75	439.92	422.04	420.38	445.36
13	NUG15	0.00	0.00	1,068.49	1,192.52	1,095.20	1,006.21	873.73	930.02	974.72	1,213.55	964.61	1,037.80
14	BA12	12,119.18	13,778.15	13,298.51	13,778.15	13,115.91	12,626.76	13,192.52	12,369.86	12,810.03	12,406.88	13,175.70	12,119.18
15	BA12TS	0.00	0.00	19,943.85	20,817.20	24,410.48	35,711.35	26,322.88	23,723.02	32,902.46	26,558.50	33,104.41	22,935.16
16	BA14	7,240.40	8,280.35	7,245.60	8,280.35	7,785.82	7,782.31	7,262.45	7,489.29	7,859.92	7,645.31	7,761.72	7,240.40
17	BA14TS	0.00	0.00	12,759.73	9,789.16	16,960.11	9,630.06	20,625.86	8,908.66	12,390.49	11,564.86	14,061.98	13,246.51
18	AB20	0.00	0.00	6,906.81	7,465.71	6,220.17	6,543.77	6,409.55	9,050.06	7,509.93	6,118.65	7,091.11	5,993.09
19	SC30	3,593.51	5,418.52	4,422.05	5,258.29	4,250.73	4,761.32	5,003.13	5,120.84	5,418.52	3,593.51	5,138.67	4,028.23
20	SC35	4,812.64	9,582.88	9,582.88	6,335.04	6,511.53	7,013.25	7,169.63	7,739.58	6,675.96	4,812.64	7,477.18	7,962.63

Tabel 4.6 Data Kelayakan Setiap Solusi Untuk Dua Puluh Kasus

No	Problem Name	1	2	3	4	5	6	7	8	9	10
1	O7										
2	FO7										
3	FO8										
4	O9										
5	V10s										
6	V10a										
7	M11s										
8	M11a										
9	M15s										
10	M15a										
11	M25										
12	NUG12										
13	NUG15										
14	BA12										
15	BA12TS										
16	BA14										
17	BA14TS										
18	AB20										
19	SC30										
20	SC35										

4.4 Analisis

Tabel 4.7 Tabel Perbandingan Antara Metode-metode

No	Problem Set	Tate and Smith (1995)	Konak et al. (2006)	Liu and Meller (2007)	Scholz et al. (2009)	Komarudin (2009)	Nilai objektif terbaik algoritma DE
1	O7	-	-	131.63	132.00	136.58	151.1
2	FO7	-	23.12	20.73	-	23.12	27.42
3	FO8	-	22.39	22.31	-	22.39	34.81
4	O9	-	241.06	235.95	239.07	241.06	292.23
5	V10s	-	22,899.65	19,997.00	19,994.10	22,899.64	34809.68
6	V10a	-	21,463.07	-	-	21,463.10	28367.79
7	M11s**	-	1,317.79	-	-	1,321.35	-
8	M11a	-	1,225.00	-	-	1,204.15	1727.78
9	M15s	-	27,781.95	-	-	23,197.80	60621.17
10	M15a**	-	31,779.09	-	-	27,545.30	-
11	M25**	-	-	-	-	1,496.42	-
12	NUG12	-	265.60	-	-	262.00	420.38
13	NUG15	-	526.75	-	-	536.75	0
14	BA12	-	8,801.33	8,702.00	8,264.00	8,786.00	12119.18
15	BA12*	-	8,801.33	8,702.00	8,264.00	8,299.50	-
16	BA12TS**	8,861.00	8,600.33	-	-	8,587.05	-
17	BA14	-	5,004.55	5,004.00	4,712.33	5,004.55	7240.4
18	BA14*	-	5,004.55	5,004.00	4,712.33	4,913.22	-
19	BA14TS**	5,080.10	4,927.69	-	-	4,927.69	-
20	AB20**	7,205.40	6,890.82	-	-	5,677.83	-
21	SC30*	-	-	3,706.83	-	3,679.85	3593.5076
22	SC35*	-	-	3,604.12	-	3,962.72	4812.64

* nilai yang didapat oleh *Ant System* menggunakan mFBS (Komarudin, 2009)

** tidak ditemukan solusi yang layak

Dari **Tabel 4.7** dapat dilihat jika dibandingkan dengan kelima metode yang sudah dikembangkan mendahului algoritma DE, algoritma ini hampir selalu tidak dapat menyamai nilai optimal yang ditemukan oleh metode-metode lainnya. Hanya pada problem SC30 saja metode ini mampu menemukan desain optimal yang lebih baik dibandingkan dengan metode yang dikembangkan oleh Tate dan Smith (1995), Konak *et al* (2006), dan Komarudin (2009).

Pada beberapa kasus, nilai terbaik yang ditemukan oleh algoritma DE seperti untuk kasus M11s sesungguhnya ditemukan dan dikenakan biaya penalti. Tetapi karena ketentuan kelayakan yang sudah dijelaskan pada bab 2, maka tidak ada satupun desain yang dapat dianggap layak. Hal ini dapat dikarenakan terlalu banyak departemen yang harus diperhitungkan oleh algoritma DE, sehingga perubahan variabel-variabel pada

satu elemen pasti akan diikuti oleh perubahan variabel elemen lainnya, sehingga sulit untuk mendapatkan desain yang lebih baik. Lagipula tidak tersedia ruang kosong yang lebih pada fasilitas untuk menggerakkan departemen-departemen ini, sehingga semakin kecil kemungkinan seluruh departemen tidak beririsan.

Pada **Tabel 4.6** dapat kita lihat pula, bahwa pada problem O7, FO7, hingga problem V10a, algoritma DE hanya beberapa kali menemukan solusi yang tidak layak. Akan tetapi ketika jumlah departemen meningkat, algoritma DE kehilangan kemampuannya untuk menemukan solusi yang layak.

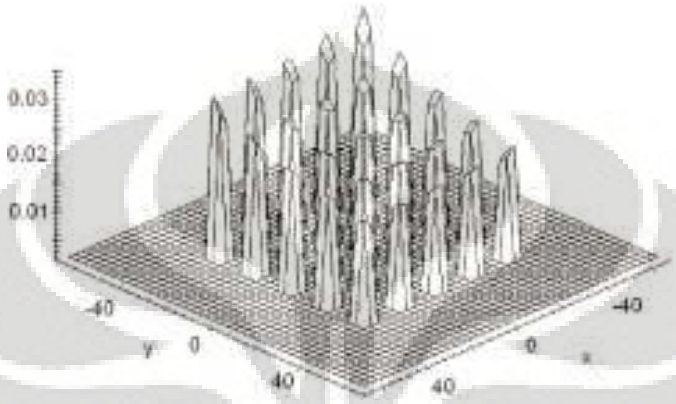
Jika kita melihat lebih dalam lagi, kasus BA12 dengan BA12TS maupun BA14 dengan BA14TS adalah dua pasang perbandingan kasus yang menarik. Kasus BA12 adalah kasus dengan ruang kosong, sedangkan BA12TS adalah kasus tanpa ruang kosong. Pada **Tabel 4.6** dapat kita lihat bahwa terdapat perbedaan mencolok dari kinerja algoritma DE, yang mana pada kasus BA12 dengan ruang kosong, semua solusi yang didapat algoritma DE adalah solusi yang layak; bertolak belakang dengan kasus BA12TS tanpa ruang kosong tidak ada satupun solusi yang layak berhasil dihasilkan.

Nilai standar deviasi pada beberapa problem yang kecil pada **Tabel 4.2** dapat terjadi dikarenakan nilai arus material yang terlibat pada problem ini homogen, sesuai dengan karakteristik problem yang sudah dijelaskan sebelumnya. Yang mana menyebabkan nilai standar deviasi tidak lagi dapat dijadikan acuan konsistensi dari algoritma DE ini.

Nilai optimal yang tidak konsisten seperti yang sudah dijelaskan pada bagian sebelumnya dapat disamakan dengan lemahnya tingkat konvergensi algoritma DE pada UA-FLP ini, walaupun secara umum penelitian yang dilakukan oleh Storn dan Price (1997) menyebutkan bahwa algoritma DE memiliki tingkat konvergensi yang bagus. Sejauh ini penulis baru memiliki sejumlah hipotesa mengenai penyebab-penyebab tingkat konvergensi yang buruk yang terjadi ketika algoritma DE diaplikasikan pada UA-FLP ini, antara lain:

a) Penerapan nilai penalti yang rumit menyebabkan kontur area-area pencarian menjadi amat ekstrim, yang mana antara area optimum yang satu dengan area optimum yang lain dipisahkan oleh nilai-nilai penalti yang tinggi, sehingga algoritma DE kehilangan keunggulannya dalam pengacakan vektor. Hipotesa ini

sulit penulis buktikan karena terdapat tiga variabel yang diproses oleh algoritma, sehingga sulit untuk mengilustrasikannya dalam gambar.



Gambar 4.4 Fungsi Foxholes (Karaboğa dan Ökdem, 2004)

Meskipun demikian, **Gambar 4.2** memberikan ilustrasi yang baik mengenai kasus UA-FLP ini. Setiap pasak menggambarkan desain-desain optimal, dan antara setiap pasak tersebut terdapat jarak yang lebarnya untuk kasus UA-FLP ini kemungkinan cukup besar sehingga algoritma DE yang mengandalkan nilai differensial sebagai alat pencari tidak mampu berpindah dari satu pasak ke pasak yang lainnya. Jika algoritma tidak mampu berpindah dari satu pasak ke pasak yang lainnya, tidak aneh jika algoritma DE ini memiliki tingkat konsistensi yang rendah, karena algoritma ini akan cenderung terjebak pada satu *local optima*. Sebagai bukti bahwa algoritma DE terjebak pada satu *local optima* adalah setiap kali desain optimal (untuk suatu *local* optima) sudah tercapai, nilai optimal juga macet pada nilai tersebut juga.

b) Metode perumusan fungsi yang mengubah vektor menjadi sebuah variabel masih kurang bagus. Rumus (4.6) masih memerlukan perbaikan dan penelitian yang lebih mendalam.

c) Nilai dari F yang dipilih masih kurang tepat, sehingga membatasi kemampuan pencarian dari algoritma DE (Karaboğa dan Ökdem, 2004). Penelitian ini memang masih membatasi pada nilai F antara 0 hingga 1. Dibatasi demikian karena akan membutuhkan waktu lebih lama lagi untuk mengetahui nilai F yang tepat untuk masing-masing problem, terutama karena ukuran fasilitas yang bervariasi tentunya

menyebabkan area pencarian juga makin bervariasi. Lebih jauh, dikarenakan pada algoritma DE ketika diaplikasikan pada UA-FLP pergerakannya amat bergantung pada vektor, menurut pengamatan penulis ada kaitan antara rasio ukuran departemen dengan ukuran fasilitas. Setiap problem yang rasio fasilitasnya sangat besar, seperti pada problem AB20 cenderung lebih stabil dan lebih cepat dalam menghasilkan desain optimal.

BAB 5
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Dari penelitian dan analisis yang sudah penulis lakukan pada bab 4, dapat diambil beberapa kesimpulan.

Kesimpulan yang pertama adalah algoritma DE dapat diimplementasikan pada UA-FLP untuk mendapatkan nilai solusi dan desain solusinya.

Kesimpulan kedua adalah Algoritma DE ketika diimplementasikan pada UA-FLP masih memiliki banyak kekurangan, yang terutama adalah masih tidak konsisten dalam menemukan nilai solusinya.

Kesimpulan yang ketiga adalah algoritma DE bekerja lebih baik pada problem-problem yang memiliki ruang lebih, sehingga kejadian antar departemen beririsan lebih kecil. Pada kasus di atas 10 departemen, algoritma DE baru bisa menghasilkan solusi jika ada ruang kosong pada fasilitas

Kesimpulan yang terakhir adalah algoritma DE hanya mampu menemukan nilai optimal baru untuk problem SC30.

5.2 Saran

Dari hasil analisis pada bab 4, ada beberapa saran untuk pengembangan lebih lanjut dari implementasi algoritma DE pada UA-FLP ini. Saran-saran tersebut antara lain:

- Melakukan penelitian lebih jauh mengenai nilai F yang tepat untuk masing-masing problem, dan bagaimana metode menentukan nilai F yang tepat. Karena besarnya nilai F yang dibutuhkan tidak berbanding lurus dengan besarnya jumlah departemen pada problem.
- Memperbaiki aturan pemberian nilai penalti untuk setiap kejadian yang harus dikenakan nilai penalti.
- Melakukan penelitian lebih mendalam mengenai penyebab problem konvergensi yang dialami oleh algoritma DE ketika diimplementasikan pada UA-FLP.

- Menambahkan algoritma-algoritma tambahan yang berfungsi sebagai *local search* atau algoritma untuk mengeksplorasi dengan baik satu wilayah pencarian.

DAFTAR REFERENSI

Bhowmik, R. (2008). An Approach to the Facility Layout Design Optimization. *IJCSNS International Journal of Computer Science and Network Security*, 8, 4.

Engelbrecht, P.A. (2007). *Computational Intelligence*: *An Introduction* (2nd ed). South Africa: John Wiley & Sons, Ltd

Garson,D.G. (2010). *Reliability Analysis*. 30 Januari, 2010. http://faculty.chass.ncsu.edu/garson/PA765/reliab.htm

Karaboga, D. & Okdem, S. (2004). A simple and global optimization algorithm for engineering problems: Differential Evolution algorithm. *Turkey Journal Engineering*, 12, 1.

Komarudin. 2009. An Improved Ant System Algorithm For Unequal Area Facility Layout Problems. Tesis akses bebas. Skudai: Universiti Teknologi Malaysia.

Konak, A., Kulturel-Konak, S., Norman, B.A., and Smith, A.E. (2006). A New Mixed Integer Formulation for Optimal Facility Layout design. *Operation Research Letters.* 34(6), 660-672.

Lampinen, J. (2002). A Constraint Handling Approach for the Differential Evolution Algorithm. *Proceedings of the Congress on Evolutionary Computation 2002 (CEC'2002),* 2, 1468-1473. Piscataway, New Jersey: IEEE Service Center.

Liu, Q., and Meller, R.D. (2007). A sequence-pair representation and MIP-modelbased heuristic for the facility layout problem with rectangular departments. *IIE Transactions*. 39(4), 377-394.

Luke, S. (2010). Essentials of Metaheuristics. Virginia: George Mason University.

Scholz, D., Petrick, A., and Domschke, W. (2009). STaTS: A slicing tree and tabu search based heuristic for the unequal area facility layout problem. *European Journal of Operational Research*. 197(1), 166-178.

Storn, R., Price, K. (1997). Differential Evolution - A simple and efficient adaptive scheme for global optimization over continuous spaces. *Journal of Global Optimization*, 11, 341–359.

LAMPIRAN

1.1 Ringkasan Dari 20 Set Data UA-FLP

No	Problem Sets	Number of Departments	Facility Size			Common Shape Constraints (α max)
			Width	Height	Area	
1	O7	7	8.54	13.00	111.02	α max = 4
2	FO7	7	8.54	13.00	111.02	α max = 5
3	FO8	8	11.31	13.00	147.03	α max = 5
4	O9	9	12.00	13.00	156.00	α max = 5
5	V10s	10	25.00	51.00	1,275.00	l min = 5
6	V10a	10	25.00	51.00	1,275.00	α max = 5
7	M11s	11	6.00	6.00	36.00	l min = 1
8	M11a	11	3.00	2.00	6.00	α max = 5
9	M15s	15	15.00	15.00	225.00	l min = 1
10	M15a	15	15.00	15.00	225.00	α max = 5
11	M25	25	15.00	15.00	225.00	α max = 5
12	NUG12	12	3.00	4.00	12.00	α max = 5
13	NUG15	15	3.00	5.00	15.00	α max = 5
14	BA12	12	7.00	9.00	63.00	l min = 1
15	BA12TS	16	7.00	9.00	63.00	l min = 1
16	BA14	14	6.00	10.00	60.00	l min = 1
17	BA14TS	14	6.00	10.00	60.00	l min = 1
18	AB20	20	2.00	3.00	6.00	α max = 1.75
19	SC30	30	12.00	15.00	180.00	α max = 5
20	SC35	35	15.00	16.00	240.00	α max = 4

1.2 Problem O7

Area Requirement

Dept	Area
1	16
2	16
3	16
4	36
5	9
6	9
7	9

Material Flow

From	To	Material Flow
1	4	5
1	7	1
2	4	3
2	7	1
3	4	2
3	7	1
4	5	4
4	6	4
5	7	2
6	7	1

1.3 Problem FO7

Area Requirement

Dept	Area
1	16
2	16
3	16
4	36
5	9
6	9
7	9

Material Flow

From	To	Material Flow
1	2	1
2	3	1
3	4	1
4	5	1
5	6	1
6	7	1

1.4 Problem FO8

Area Requirement

Dept	Area
1	16
2	16
3	16
4	36
5	36
6	9
7	9
8	9

Material Flow

From	To	Material Flow
1	2	1
2	3	1
3	4	1
4	5	1
5	6	1
6	7	1
7	8	1

1.5 Problem O9

Area Requirement

Dept	Area
1	16
2	16
3	16
4	36
5	36
6	9
7	9
8	9
9	9

Material Flow

From	To	Material Flow
1	4	5
1	5	5
1	9	1
2	4	3
2	5	3
2	9	1
3	4	2
3	5	2
3	9	1
4	6	4
4	7	4
5	6	3
5	9	4
6	9	2
7	9	1

1.6 Problem V10s

Dept	1	2	3	4	5	6	7	8	9	10	Area	Min side
1	-					218					238	1
2		-				148			296		112	1
3			-	28	70						160	1
4				-		28	70	140			80	1
5					-			210			120	1
6						-					80	1
7							-			28	60	1
8								-		888	85	1
9									-	59.2	221	1
10										-	119	1

1.7 Problem V10a

Dept	1	2	3	4	5	6	7	8	9	10	Area	Max Aspect Ratio
1	-	0	0	0	0	218	0	0	0	0	238	5
2		-	0	0	0	148	0	0	296	0	112	5
3			-	28	70	0	0	0	0	0	160	5
4				-	0	28	70	140	0	0	80	5
5					-	0	0	210	0	0	120	5
6						-	0	0	0	0	80	5
7							-	0	0	28	60	5
8								-	0	888	85	5
9									-	59.2	221	5
10										-	119	5

1.8 Problem M11s

Dept	1	2	3	4	5	6	7	8	9	10	11	Area	Min side
1	-	10			140	90	20		40			3	1
2		-	10									2	1
3			-	10								4	1
4				-							4	5	1
5		10			-		40			20		2	1
6			10			-			20			3	1
7							-	10				4	1
8								-			11	5	1
9									-	20		1	1
10										-	20	1	1
11	146										-	6	1

1.9 Problem M11a

Dept	1	2	3	4	5	6	7	8	9	10	11	Area	Max Aspect Ratio
1	0	10	0	0	140	90	20	0	40	0	0	3	5
2	0	0	10	0	0	0	0	0	0	0	0	2	5
3	0	0	0	10	0	0	0	0	0	0	0	4	5
4	0	0	0	0	0	0	0	0	0	0	4	5	5
5	0	10	0	0	0	0	40	0	0	20	0	2	5
6	0	0	10	0	0	0	0	0	20	0	0	3	5
7	0	0	0	0	0	0	0	10	0	0	0	4	5
8	0	0	0	0	0	0	0	0	0	0	11	5	5
9	0	0	0	0	0	0	0	0	0	20	0	1	5
10	0	0	0	0	0	0	0	0	0	0	20	1	5
11	146	0	0	0	0	0	0	0	0	0	0	6	5

1.10 Problem M15s

Dept	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Area	Min side
1															240	15	1
2	240															10	1
3				1200												9	1
4										1200						7	1
5														600		9	1
6								480								25	1
7								480								25	1
8															120	15	1
9										600						10	1
10												600				25	1
11							480									10	1
12															600	15	1
13							480									6	1
14												600				19	1
15		10	50		25	40			25		40		20			25	1

1.11 Problem M15a

Dept	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Area	Max aspect ratio
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	240	15	5
2	240	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	5
3	0	0	0	1200	0	0	0	0	0	0	0	0	0	0	0	9	5
4	0	0	0	0	0	0	0	0	0	1200	0	0	0	0	0	7	5
5	0	0	0	0	0	0	0	0	0	0	0	0	0	600	0	9	5
6	0	0	0	0	0	0	0	480	0	0	0	0	0	0	0	25	5
7	0	0	0	0	0	0	0	480	0	0	0	0	0	0	0	25	5
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	120	15	5
9	0	0	0	0	0	0	0	0	0	600	0	0	0	0	0	10	5
10	0	0	0	0	0	0	0	0	0	0	0	600	0	0	0	25	5
11	0	0	0	0	0	0	480	0	0	0	0	0	0	0	0	10	5
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	600	15	5
13	0	0	0	0	0	0	480	0	0	0	0	0	0	0	0	6	5
14	0	0	0	0	0	0	0	0	0	0	0	600	0	0	0	19	5
15	0	10	50	0	25	40	0	0	25	0	40	0	20	0	0	25	5

1.12 Problem M25

Dept	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	Area	Max aspect ratio
1	0	0	0	20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	5
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	1	5
3	0	0	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	5
4	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	5
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	1	5
6	0	0	0	0	0	0	0	0	40	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	5
7	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	5	5
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	3	5
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	40	0	0	0	0	0	0	2	5
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	1	5
11	0	0	0	0	0	0	0	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	5
12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	40	0	0	0	0	0	0	0	0	2	5
13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	4	5
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	40	0	0	0	0	0	0	0	5	5
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	80	0	0	0	0	0	2	5
16	0	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	5
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	40	5	0	0	1	5
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	20	0	0	0	5	5
19	0	0	0	0	0	0	0	0	0	20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	5
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	40	0	0	0	4	5
21	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	5
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	16	4	5
23	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	5	5
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	2	5
25	4	5	4	0	0	16	0	4	0	0	0	16	0	8	8	8	0	0	0	0	0	0	0	0	0	4	5

1.13 Problem NUG12

Dept	1	2	3	4	5	6	7	8	9	10	11	12	Area	Max Aspect Ratio
1		5	2	4	1			6	2	1	1	1	1	5
2			3		2	2	2		4	5			1	5
3								5	5	2	2	2	1	5
4					5	2	2	10			5	5	1	5
5						10				5	1	1	1	5
6							5	1	1	5	4		1	5
7								10	5	2	3	3	1	5
8											5		1	5
9											10	10	1	5
10											5		1	5
11												2	1	5
12													1	5

1.14 Problem NUG15

From	To	Material Flow	From	To	Material Flow	From	To	Material Flow
1	2	10	3	11	2	6	14	5
1	4	5	3	12	2	6	15	10
1	5	1	3	13	5	7	8	6
1	7	1	3	14	5	7	10	1
1	8	2	3	15	5	7	11	5
1	9	2	4	5	1	7	12	5
1	10	2	4	6	1	7	13	5
1	11	2	4	7	5	7	14	1
1	13	4	4	10	2	8	9	5
2	3	1	4	11	1	8	10	2
2	4	3	4	13	2	8	11	10
2	5	2	4	14	5	8	13	5
2	6	2	5	6	3	9	11	10
2	7	2	5	7	5	9	12	5
2	8	3	5	8	5	9	13	10
2	9	2	5	9	5	9	15	2
2	11	2	5	10	1	10	12	4
2	13	10	5	12	3	10	15	5
2	14	5	5	14	5	11	12	5
3	4	10	5	15	5	11	14	5
3	5	2	6	7	2	12	13	3
3	7	2	6	8	2	12	14	3
3	8	5	6	9	1	13	14	10
3	9	4	6	10	5	13	15	2
3	10	5	6	13	2	14	15	4

Dept	Area	Max aspect Ratio
1	1	5
2	1	5
3	1	5
4	1	5
5	1	5
6	1	5
7	1	5
8	1	5
9	1	5
10	1	5
11	1	5
12	1	5
13	1	5
14	1	5
15	1	5

1.15 Problem BA12

Dept	Area	Min Side
1	9	1
2	8	1
3	10	1
4	6	1
5	4	1
6	3	1
7	3	1
8	4	1
9	2	1
10	2	1
11	1	1
12	1	1

Dept	1	2	3	4	5	6	7	8	9	10	11	12
1	-	288	180	54	72	180	27	72	36	0	0	9
2		-	240	54	72	24	48	160	16	64	8	16
3			-	120	80	0	60	120	60	0	0	30
4				-	72	18	18	48	24	48	12	0
5					-	12	12	64	16	16	4	8
6						-	18	24	6	12	3	3
7							-	0	6	6	3	6
8								-	16	16	16	4
9									-	4	4	2
10										-	2	2
11											-	2
12												-

1.16 Problem BA12TS

Dept	Area	Min Side
1	9	1
2	8	1
3	10	1
4	6	1
5	4	1
6	3	1
7	3	1
8	4	1
9	2	1
10	2	1
11	2	1
12	2	1
13	2	0
14	1	0
15	1	0
16	1	0

Dept	1	2	3	4	5	6	7	8	9	10	11	12
1	-	288	180	54	72	180	27	72	36	0	0	9
2		-	240	54	72	24	48	160	16	64	8	16
3			-	120	80	0	60	120	60	0	0	30
4				-	72	18	18	48	24	48	12	0
5					-	12	12	64	16	16	4	8
6						-	18	24	6	12	3	3
7							-	0	6	6	3	6
8								-	16	16	16	4
9									-	4	4	2
10										-	2	2
11											-	2
12												-

1.17 Problem BA14

Dept	Area	Min Side
1	9	1
2	8	1
3	9	1
4	10	1
5	6	1
6	3	1
7	3	1
8	3	1
9	2	1
10	3	1
11	2	1
12	1	1
13	1	0
14	1	0

Dept	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	-	72	162	90	108	27	0	0	18	27	18	0	0	0
2		-	72	80	0	48	0	48	32	0	16	8	0	0
3			-	45	54	27	27	27	0	27	0	9	18	0
4				-	30	0	30	30	20	0	20	10	10	0
5					-	18	0	18	12	18	24	0	0	0
6						-	9	9	0	0	6	6	6	0
7							-	9	12	9	6	3	0	0
8								-	6	9	0	3	0	0
9									-	6	4	6	2	0
10										-	6	3	6	0
11											-	2	0	0
12												-	4	0
13													-	0
14														-

1.18 Problem BA14TS

Dept	Area	Min Side
1	9	1
2	8	1
3	9	1
4	10	1
5	6	1
6	3	1
7	3	1
8	3	1
9	2	1
10	3	1
11	2	1
12	1	1
13	1	0
14	3	0

Dept	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	-	72	162	90	108	27	0	0	18	27	18	0	0	0
2		-	72	80	0	48	0	48	32	0	16	8	0	0
3			-	45	54	27	27	27	0	27	0	9	18	0
4				-	30	0	30	30	20	0	20	10	10	0
5					-	18	0	18	12	18	24	0	0	0
6						-	9	9	0	0	6	6	6	0
7							-	9	12	9	6	3	0	0
8								-	6	9	0	3	0	0
9									-	6	4	6	2	0
10										-	6	3	6	0
11											-	2	0	0
12												-	4	0
13													-	0
14														-

1.19 Problem AB20

Dept	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	Area	Max aspect ratio
1	-	18	12	0	0	0	0	0	0	10.4	11.2	0	0	12	0	0	0	0	0	0	0.27	1.75
2	18	-	9.6	244.5	7.8	0	139.5	0	12	13.5	0	0	0	0	0	0	0	0	69	0	0.18	1.75
3	12	9.6	-	0	0	22.1	0	0	31.5	39	0	0	0	130.5	0	0	0	0	136.5	0	0.27	1.75
4	0	244.5	0	-	10.8	57	75	0	23.4	0	0	14	0	0	0	0	0	15	157.5	0	0.18	1.75
5	0	7.8	0	10.8	-	0	22.5	13.5	0	15.6	0	0	0	0	13.5	0	0	0	0	0	0.18	1.75
6	0	0	22.1	57	0	-	61.5	0	0	0	0	4.5	0	0	0	0	0	10.5	0	0	0.18	1.75
7	0	139.5	0	75	22.5	61.5	-	240	0	18.7	0	0	0	9.6	0	0	0	16.5	0	37.5	0.09	1.75
8	0	0	0	0	13.5	0	240	-	0	0	0	0	6	0	0	0	0	0	75	334.5	0.09	1.75
9	0	12	31.5	23.4	0	0	0	0	-	0	0	0	0	75	0	0	75	0	0	0	0.09	1.75
10	10.4	13.5	39	0	15.6	0	18.7	0	0	-	3.6	120	0	186	19.2	0	0	0	52.5	0	0.24	1.75
11	11.2	0	0	0	0	0	0	0	0	3.6	-	22.5	0	30	9.6	225	0	0	0	0	0.6	1.75
12	0	0	0	14	0	4.5	0	0	0	120	22.5	-	0	0	16.5	0	150	0	84	0	0.42	1.75
13	0	0	0	0	0	0	0	6	0	0	0	0	-	80	10.4	60	0	0	0	0	0.18	1.75
14	12	0	130.5	0	0	0	9.6	0	75	186	30	0	80	-	97.5	0	0	9	0	0	0.24	1.75
15	0	0	0	0	13.5	0	0	0	0	19.2	9.6	16.5	10.4	97.5	-	0	52.5	0	0	0	0.27	1.75
16	0	0	0	0	0	0	0	0	0	0	0	0	60	0	0	-	120	0	0	0	0.75	1.75
17	0	0	0	0	0	0	0	0	75	0	0	150	0	0	52.5	120	-	0	75	0	0.64	1.75
18	0	0	0	15	0	10.5	16.5	0	0	0	0	0	0	9	0	0	0	-	46.5	0	0.41	1.75
19	0	69	136.5	157.5	0	0	0	75	0	52.5	0	84	0	0	0	0	75	46.5	-	0	0.27	1.75
20	0	0	0	0	0	0	37.5	334.5	0	0	0	0	0	0	0	0	0	0	0	-	0.45	1.75

1.20 Problem SC30

Dept	Area	Max Aspect Ratio	Dept	Area	Max Aspect Ratio
1	3	5	25	1	5
2	4	5	26	4	5
3	4	5	27	6	5
4	16	5	28	1	5
5	4	5	29	14	5
6	5	5	30	4	5
7	2	5	31	1	0
8	3	5	32	1	0
9	5	5	33	1	0
10	6	5	34	1	0
11	2	5	35	1	0
12	24	5	36	1	0
13	5	5	37	1	0
14	3	5	38	1	0
15	11	5	39	1	0
16	6	5	40	1	0
17	2	5	41	1	0
18	8	5	42	1	0
19	4	5	43	1	0
20	5	5	44	1	0
21	4	5	45	1	0
22	3	5	46	1	0
23	1	5	47	1	0
24	3	5			

From	To	Material Flow	From	To	Material Flow
1	24	2.95	10	12	38.03
1	25	6.32	11	12	8.84
1	26	1.26	12	13	12.97
1	27	2.11	12	14	4.67
1	28	1.26	12	15	53.42
1	30	13.91	12	16	1.83
2	6	20.38	13	15	12.97
2	19	4.5	14	15	4.67
2	20	3.63	15	4	63.95
2	21	2.93	15	18	190.74
2	22	1.29	16	15	4.58
2	23	1.43	17	15	0.76
4	3	394.11	18	4	190.74
5	11	4.09	19	29	9.18
5	12	33.09	20	29	7.4
6	5	8.92	21	29	5.97
6	7	2.07	22	29	2.63
6	8	4.84	23	29	2.92
6	10	4.56	24	29	5.9
6	12	1.83	25	29	12.65
6	17	0.61	26	29	2.53
7	8	12.97	27	29	4.22
8	9	43.23	28	29	2.53
9	11	4.76	29	4	190.13
9	12	38.47	30	29	59.64

1.21 Problem SC35

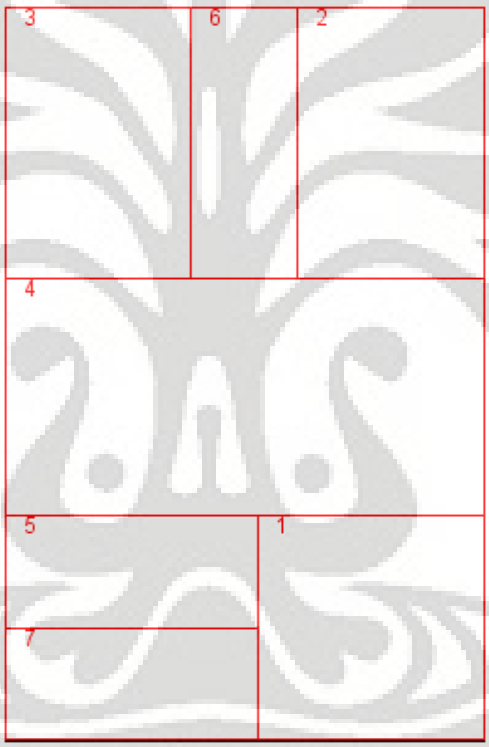
Dept	Area	Max Aspect Ratio	Dept	Area	Max Aspect Ratio
1	3	4	31	9	4
2	5	4	32	14	4
3	4	4	33	10	4
4	14	4	34	4	4
5	4	4	35	3	4
6	5	4	36	2	0
7	2	4	37	2	0
8	3	4	38	2	0
9	5	4	39	2	0
10	6	4	40	2	0
11	2	4	41	2	0
12	6	4	42	2	0
13	5	4	43	2	0
14	3	4	44	2	0
15	13	4	45	2	0
16	6	4	46	2	0
17	2	4	47	2	0
18	10	4	48	2	0
19	4	4	49	2	0
20	5	4	50	2	0
21	4	4	51	2	0
22	3	4	52	2	0
23	1	4	53	2	0
24	3	4	54	2	0
25	1	4	55	2	0
26	4	4	56	2	0
27	6	4	57	2	0
28	1	4	58	2	0
29	18	4	59	2	0
30	4	4			

From	To	Material Flow	From	To	Material Flow
1	24	2.95	11	12	8.84
1	25	6.32	12	32	72.89
1	26	1.26	13	15	12.97
1	27	2.11	14	15	4.67
1	28	1.26	15	4	63.95
1	30	13.91	15	18	190.74
2	6	20.38	16	15	4.58
2	19	4.5	17	15	0.76
2	20	3.63	18	4	190.74
2	21	2.93	19	29	9.18
2	22	1.29	20	29	7.4
2	23	1.43	21	29	5.97
4	3	225.65	22	29	2.63
5	11	4.09	23	29	2.92
5	12	33.09	24	29	5.9
6	5	8.92	25	29	12.65
6	7	2.07	26	29	2.53
6	8	4.84	27	29	4.22
6	10	4.56	28	29	2.53
6	12	1.83	29	33	190.13
6	17	0.61	30	29	59.64
6	31	3.93	31	32	22.92
7	8	12.97	32	13	12.97
8	9	43.23	32	14	4.67
9	11	4.76	32	15	53.42
9	12	38.47	32	16	1.83
10	12	38.03	33	34	168.45

2.1a Data Tata Letak Terbaik O7

Panjang	Lebar	Koordinat X	Koordinat Y
4.02	3.98	6.53	11.01
3.33	4.80	6.87	2.40
3.33	4.80	1.67	2.40
8.54	4.22	4.27	6.91
4.52	1.99	2.26	10.01
1.88	4.80	4.27	2.40
4.52	1.99	2.26	12.00

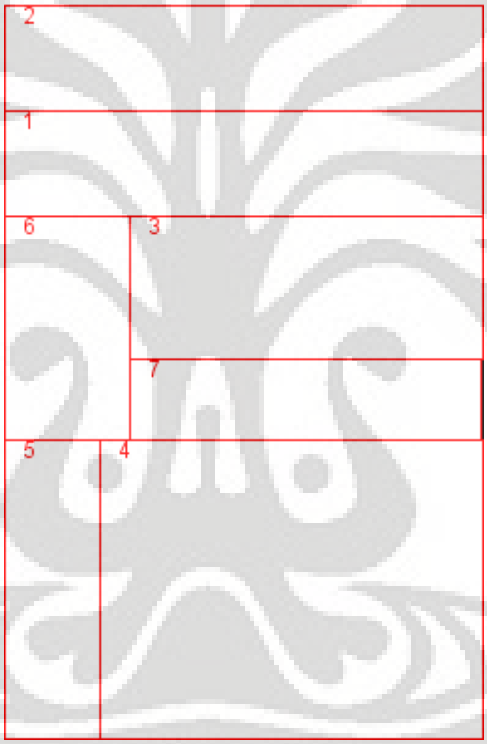
2.1b Desain Tata Letak Terbaik O7



2.2a Data Tata Letak Terbaik FO7

Panjang	Lebar	Koordinat X	Koordinat Y
8.54	1.87	4.27	2.81
8.54	1.87	4.27	0.94
6.28	2.55	5.40	5.02
6.83	5.27	5.12	10.37
1.71	5.27	0.85	10.37
2.26	3.98	1.13	5.74
6.27	1.43	5.40	7.01

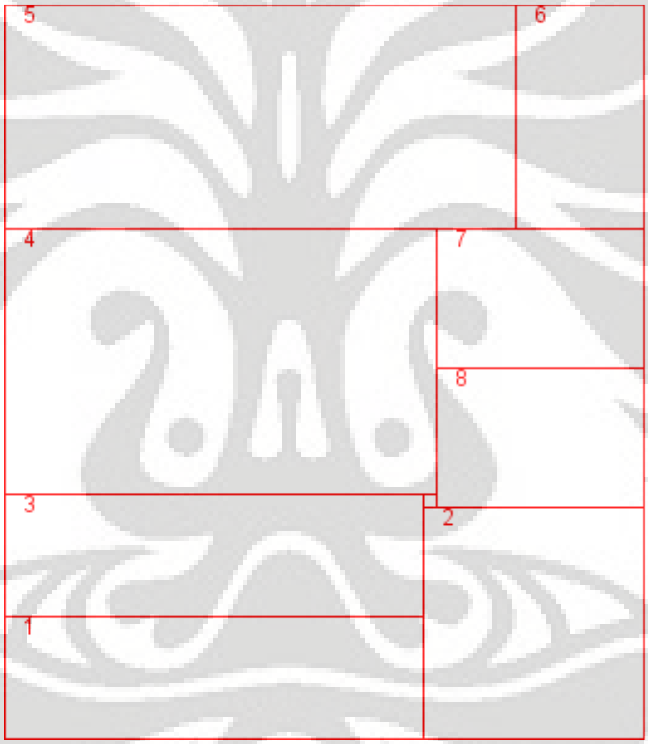
2.2b Desain Tata Letak Terbaik FO7



2.3a Data Tata Letak Terbaik FO8

Panjang	Lebar	Koordinat X	Koordinat Y
7.41	2.16	3.70	11.92
3.90	4.10	9.36	10.95
7.42	2.16	3.71	9.76
7.65	4.70	3.83	6.33
9.05	3.98	4.52	1.99
2.26	3.98	10.18	1.99
3.66	2.46	9.48	5.21
3.66	2.46	9.48	7.67

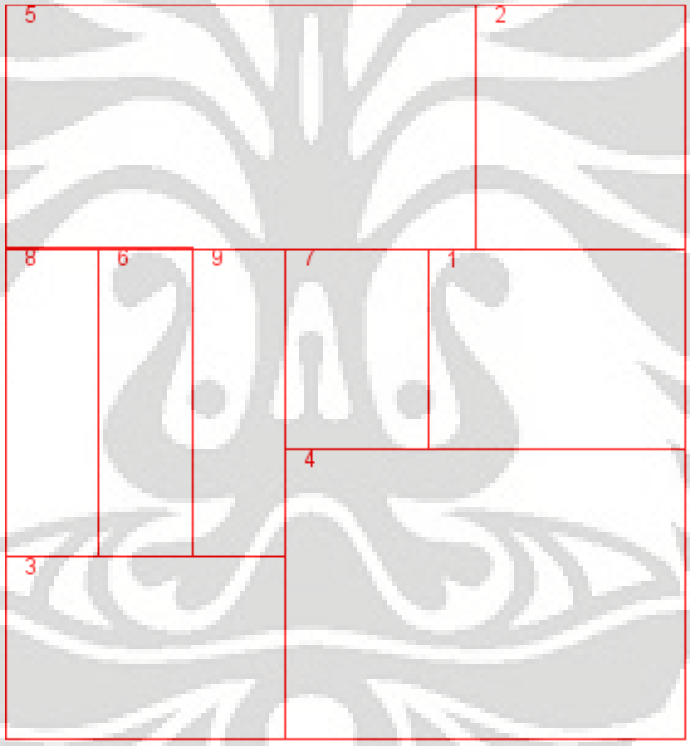
2.3b Desain Tata Letak Terbaik FO8



2.4a Data Tata Letak Terbaik O9

Panjang	Lebar	Koordinat X	Koordinat Y
4.50	3.55	9.75	6.11
3.69	4.33	10.15	2.17
4.96	3.22	2.48	11.39
7.04	5.11	8.48	10.44
8.31	4.33	4.15	2.17
1.65	5.44	2.48	7.05
2.53	3.55	6.23	6.11
1.65	5.44	0.83	7.05
1.65	5.44	4.13	7.05

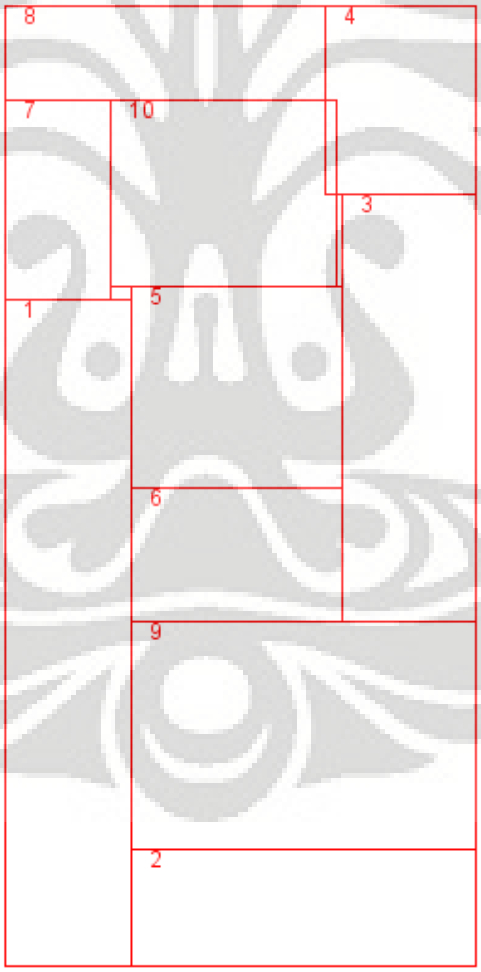
2.4b Desain Tata Letak Terbaik O9



2.5a Data Tata Letak Terbaik V10s

Panjang	Lebar	Koordinat X	Koordinat Y
6.73	35.37	3.36	33.32
18.27	6.13	15.86	47.93
7.05	22.69	21.47	21.43
7.94	10.08	21.03	5.04
11.22	10.70	12.34	20.30
11.22	7.13	12.34	29.21
5.64	10.63	2.82	10.32
17.00	5.00	8.56	2.50
18.27	12.10	15.86	38.82
11.96	9.95	11.62	9.97

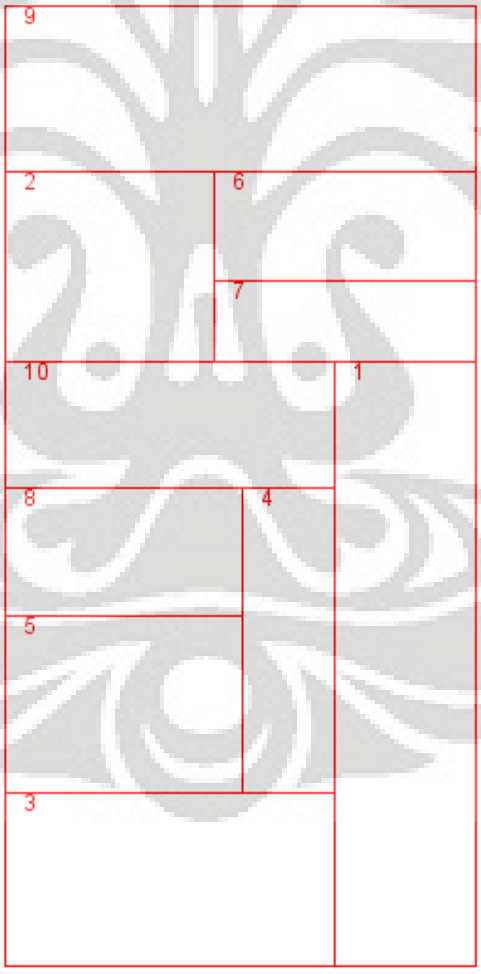
2.5b Desain Tata Letak Terbaik V10s



2.6a Data Tata Letak Terbaik V10a

Panjang	Lebar	Koordinat X	Koordinat Y
7.42	32.08	21.29	34.96
11.11	10.08	5.56	13.88
17.58	9.10	8.79	46.45
4.94	16.21	15.11	33.79
12.65	9.49	6.32	37.15
13.89	5.76	18.06	11.72
13.89	4.32	18.06	16.76
12.65	6.72	6.32	29.05
25.00	8.84	12.50	4.42
17.58	6.77	8.79	22.30

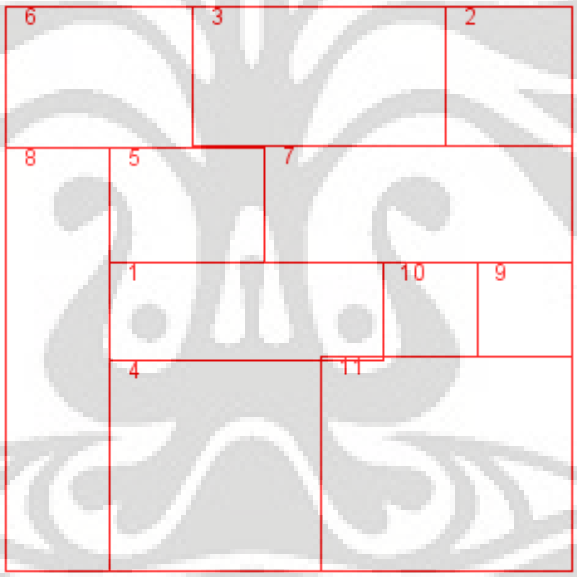
2.6b Desain Tata Letak Terbaik V10a



2.7a Data Tata Letak Terbaik M11s

Panjang	Lebar	Koordinat X	Koordinat Y
2.89	1.04	2.56	3.25
1.33	1.50	5.33	0.75
2.67	1.50	3.33	0.75
2.24	2.23	2.23	4.88
1.63	1.23	1.93	2.11
2.00	1.50	1.00	0.75
3.26	1.23	4.37	2.11
1.11	4.50	0.56	3.75
1.00	1.00	5.50	3.24
1.00	1.00	4.50	3.24
2.65	2.26	4.67	4.87

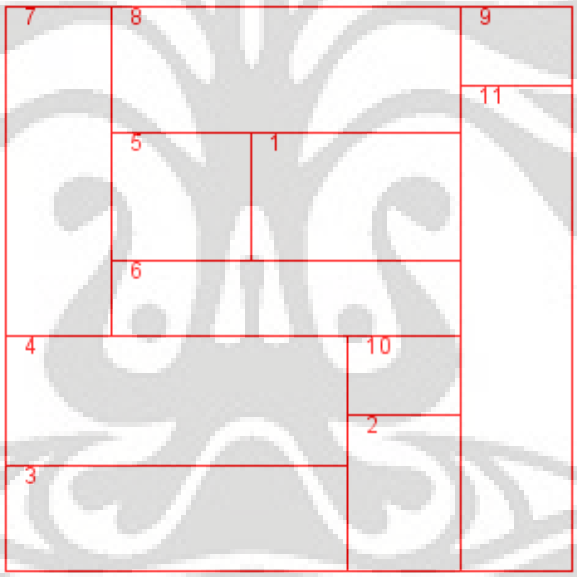
2.7b Desain Tata Letak Terbaik M11s



2.8a Data Tata Letak Terbaik M11a

Panjang	Lebar	Koordinat X	Koordinat Y
2.22	1.35	3.72	2.03
1.21	1.66	4.23	5.17
3.62	1.10	1.81	5.45
3.62	1.38	1.81	4.21
1.48	1.35	1.88	2.03
3.70	0.81	2.99	3.11
1.14	3.52	0.57	1.76
3.70	1.35	2.99	0.68
1.17	0.86	5.42	0.43
1.21	0.83	4.23	3.93
1.17	5.14	5.42	3.43

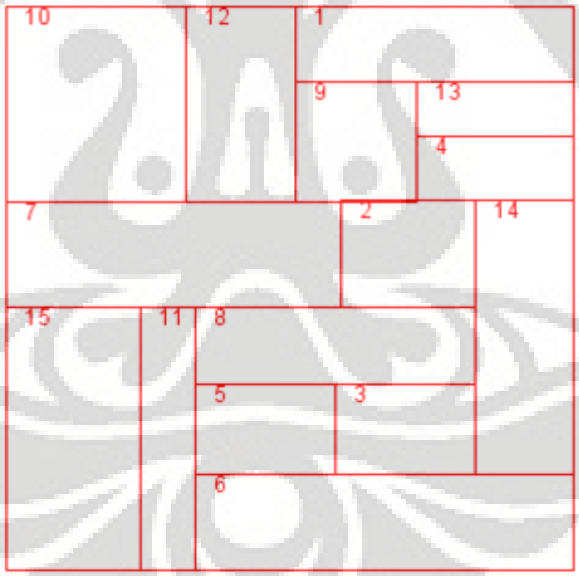
2.8b Desain Tata Letak Terbaik M11a



2.9a Data Tata Letak Terbaik M15s

Panjang	Lebar	Koordinat X	Koordinat Y
7.32	2.05	11.34	1.02
3.53	2.84	10.64	6.61
3.69	2.44	10.56	11.28
4.15	1.69	12.93	4.34
3.69	2.44	6.86	11.28
9.98	2.50	10.01	13.75
8.88	2.82	4.44	6.62
7.38	2.03	8.71	9.04
3.17	3.15	9.26	3.62
4.80	5.21	2.40	2.60
1.43	6.97	4.30	11.51
2.88	5.21	6.24	2.60
4.15	1.45	12.93	2.77
2.60	7.31	13.70	8.84
3.58	6.97	1.79	11.51

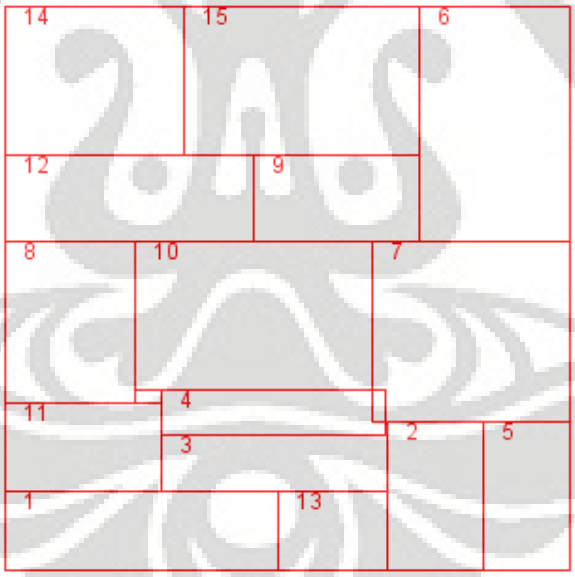
2.9b Desain Tata Letak Terbaik M15s



2.10a Data Tata Letak Terbaik M15a

Panjang	Lebar	Koordinat X	Koordinat Y
7.26	2.07	3.63	13.97
2.54	3.93	11.44	13.03
5.97	1.51	7.18	12.18
5.92	1.18	7.15	10.84
2.29	3.93	13.86	13.03
3.97	6.30	13.02	3.15
5.24	4.77	12.38	8.68
3.49	4.29	1.75	8.40
4.41	2.27	8.82	5.12
6.27	3.99	6.63	8.25
4.19	2.38	2.10	11.74
6.62	2.27	3.31	5.12
2.90	2.07	8.71	13.97
4.76	3.99	2.38	1.99
6.27	3.99	7.90	1.99

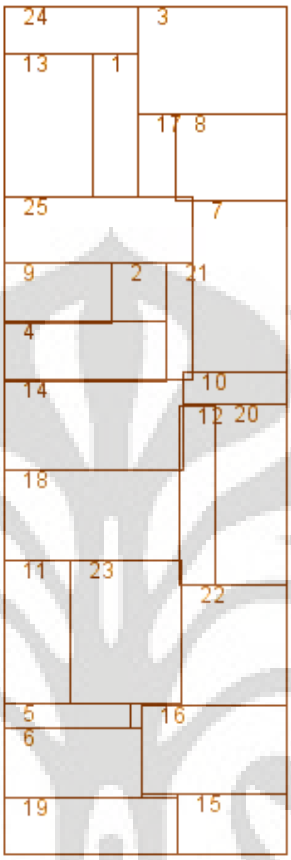
2.10b Desain Tata Letak Terbaik M15a



2.11a Data Tata Letak Terbaik M25

Panjang	Lebar	Koordinat X	Koordinat Y
0.79	2.53	1.98	2.10
0.96	1.04	2.40	5.07
2.63	1.90	3.69	0.95
2.88	1.04	1.44	6.11
2.24	0.45	1.12	12.58
2.45	1.22	1.23	13.41
1.64	3.06	4.18	4.97
1.95	1.54	4.03	2.67
1.92	1.04	0.96	5.08
1.83	0.55	4.09	6.77
1.18	2.54	0.59	11.08
0.63	3.16	3.43	8.66
1.58	2.53	0.79	2.10
3.17	1.58	1.59	7.42
1.93	1.04	4.04	14.48
2.55	1.57	3.73	13.18
0.68	1.47	2.71	2.64
3.11	1.61	1.56	9.01
3.07	0.98	1.54	14.51
1.25	3.19	4.37	8.64
0.48	2.06	3.12	5.59
1.86	2.15	4.07	11.32
1.96	2.55	2.16	11.09
2.38	0.84	1.19	0.42
3.36	1.19	1.68	3.96

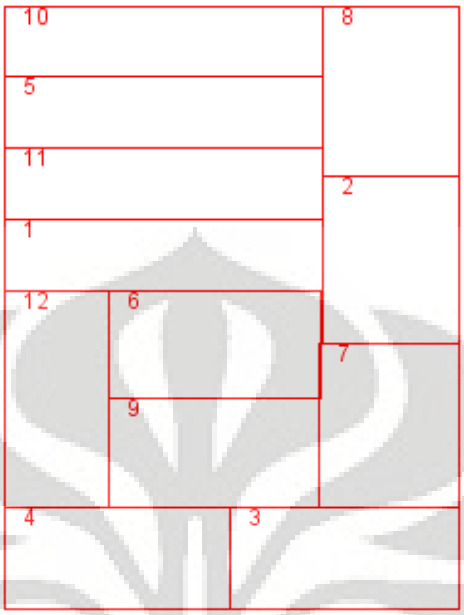
2.11b Desain Tata Letak Terbaik M25



2.12a Data Tata Letak Terbaik NUG12

Panjang	Lebar	Koordinat X	Koordinat Y
2.10	0.48	1.05	1.66
0.90	1.12	2.55	1.68
1.50	0.67	2.25	3.67
1.50	0.67	0.75	3.67
2.11	0.47	1.06	0.71
1.40	0.71	1.40	2.25
0.92	1.09	2.54	2.79
0.89	1.13	2.56	0.56
1.39	0.72	1.39	2.97
2.11	0.47	1.06	0.24
2.11	0.48	1.05	1.18
0.70	1.43	0.35	2.61

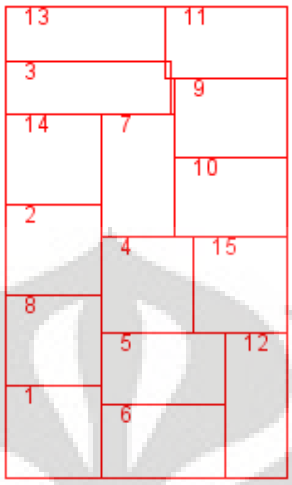
2.12b Desain Tata Letak Terbaik NUG12



2.13a Data Tata Letak Terbaik NUG15

Panjang	Lebar	Koordinat X	Koordinat Y
1.04	0.96	0.52	4.52
1.04	0.96	0.52	2.59
1.77	0.56	0.89	0.87
0.98	1.02	1.53	2.96
1.31	0.76	1.69	3.85
1.31	0.77	1.69	4.62
0.77	1.30	1.42	1.80
1.04	0.96	0.52	3.56
1.19	0.84	2.40	1.19
1.19	0.84	2.40	2.03
1.29	0.77	2.35	0.39
0.65	1.53	2.67	4.24
1.71	0.59	0.85	0.29
1.04	0.96	0.52	1.63
0.98	1.02	2.51	2.96

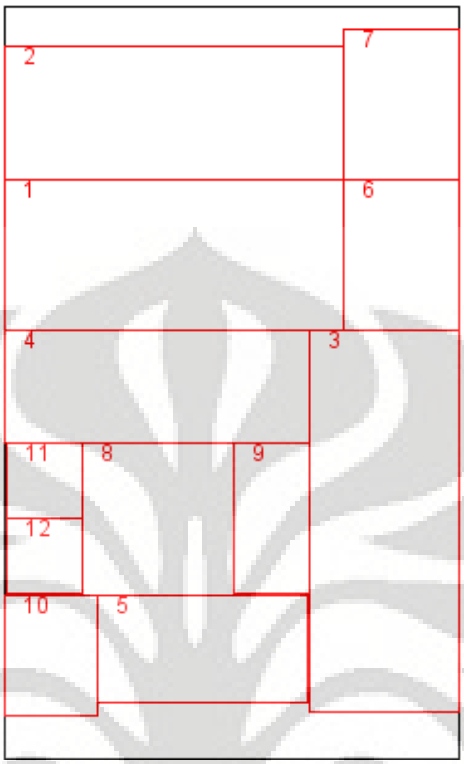
2.13b Desain Tata Letak Terbaik NUG15



2.14a Data Tata Letak Terbaik BA12

Panjang	Lebar	Koordinat X	Koordinat Y
4.50	2.00	2.25	3.32
4.50	1.78	2.25	1.43
1.97	5.07	5.01	6.86
4.02	1.49	2.02	5.07
2.78	1.44	2.64	8.55
1.50	2.00	5.25	3.32
1.50	2.00	5.25	1.32
1.99	2.01	2.04	6.82
1.00	2.00	3.53	6.82
1.25	1.60	0.62	8.63
1.00	1.00	0.54	6.31
1.00	1.00	0.54	7.31

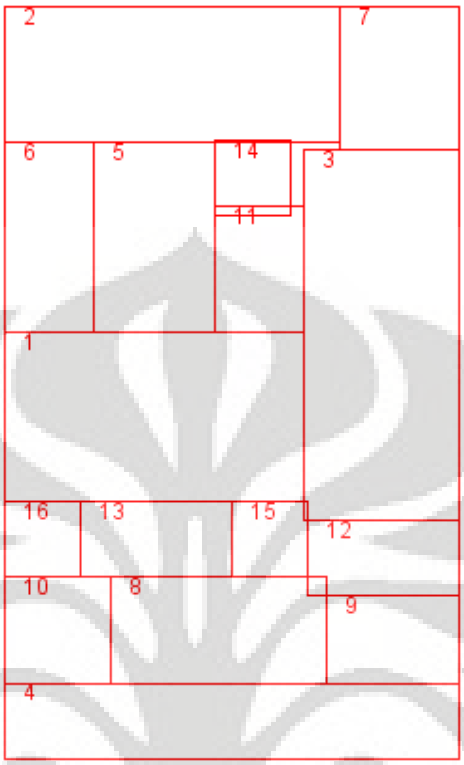
2.14b Desain Tata Letak Terbaik BA12



2.15a Data Tata Letak Terbaik BA12TS

Panjang	Lebar	Koordinat X	Koordinat Y
3.97	2.26	1.99	5.46
4.43	1.80	2.22	0.90
2.03	4.94	4.99	4.38
6.00	1.00	3.00	9.50
1.59	2.52	1.98	3.07
1.19	2.52	0.59	3.07
1.57	1.91	5.22	0.96
2.84	1.41	2.84	8.30
1.74	1.15	5.13	8.42
1.42	1.41	0.71	8.30
1.20	1.67	3.37	3.49
2.00	1.00	5.00	7.35
2.00	1.00	2.00	7.09
1.00	1.00	3.28	2.30
1.00	1.00	3.50	7.09
1.00	1.00	0.50	7.09

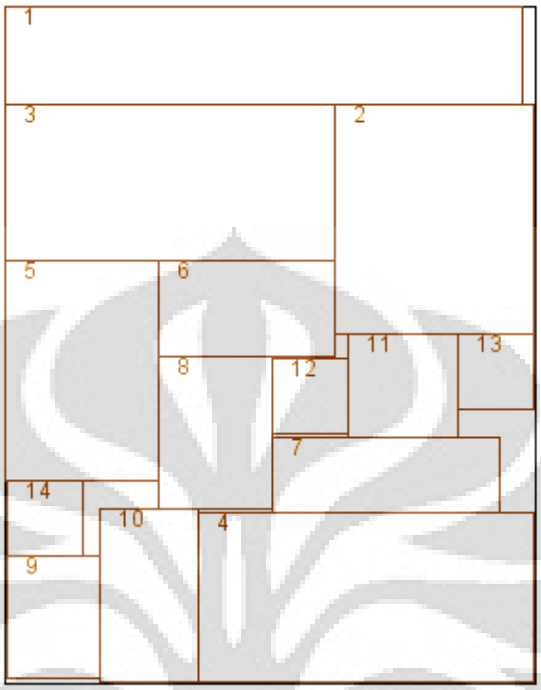
2.15b Desain Tata Letak Terbaik BA12TS



2.15a Data Tata Letak Terbaik BA14

Panjang	Lebar	Koordinat X	Koordinat Y
6.83	1.32	3.41	0.66
2.62	3.05	5.68	2.84
4.36	2.06	2.19	2.35
4.45	2.25	4.78	7.87
2.05	2.93	1.02	4.85
2.32	1.29	3.21	4.03
3.00	1.00	5.04	6.24
1.49	2.01	2.79	5.68
1.23	1.63	0.64	8.13
1.30	2.31	1.90	7.84
1.46	1.37	5.27	5.06
1.00	1.00	4.04	5.18
1.00	1.00	6.49	4.87
1.00	1.00	0.54	6.81

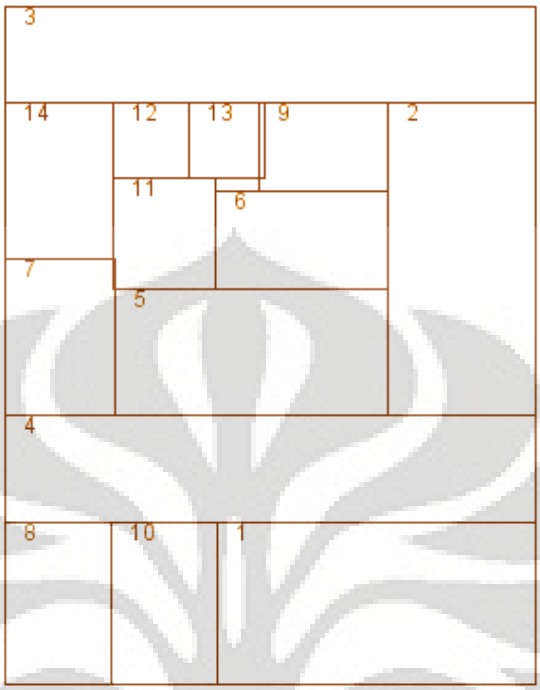
2.15b Desain Tata Letak Terbaik BA14



2.15a Data Tata Letak Terbaik BA14TS

Panjang	Lebar	Koordinat X	Koordinat Y
4.20	2.14	4.90	7.93
1.93	4.14	6.03	3.36
7.00	1.29	3.50	0.64
7.00	1.43	3.50	6.14
3.61	1.66	3.26	4.60
2.28	1.32	3.93	3.11
1.46	2.06	0.73	4.40
1.40	2.14	0.70	7.93
1.72	1.17	4.21	1.87
1.40	2.14	2.10	7.93
1.35	1.48	2.11	3.03
1.00	1.00	1.94	1.79
1.00	1.00	2.94	1.79
1.44	2.09	0.72	2.33

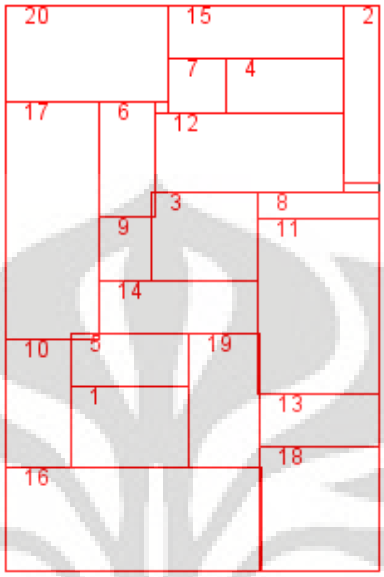
2.15b Desain Tata Letak Terbaik BA14TS



2.15a Data Tata Letak Terbaik AB20

Panjang	Lebar	Koordinat X	Koordinat Y
0.63	0.43	0.67	2.24
0.19	0.95	1.91	0.47
0.58	0.47	1.07	1.23
0.63	0.29	1.50	0.43
0.63	0.28	0.67	1.88
0.29	0.62	0.65	0.82
0.31	0.29	1.03	0.43
0.64	0.14	1.68	1.06
0.27	0.33	0.64	1.29
0.35	0.68	0.18	2.11
0.64	0.94	1.68	1.60
1.01	0.42	1.31	0.79
0.63	0.28	1.68	2.21
0.85	0.28	0.93	1.60
0.93	0.29	1.34	0.14
1.37	0.55	0.69	2.73
0.51	1.26	0.25	1.14
0.63	0.65	1.68	2.67
0.38	0.71	1.18	2.10
0.88	0.51	0.44	0.26

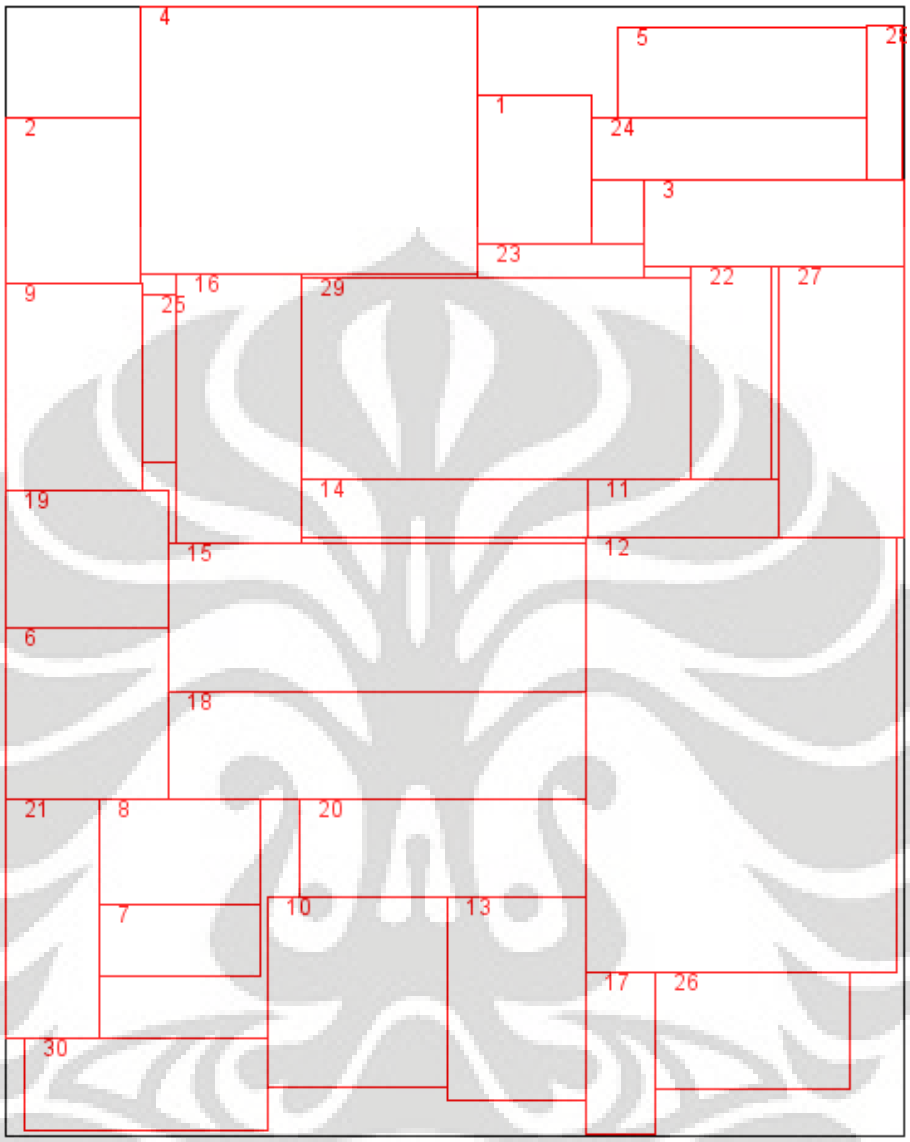
2.15b Desain Tata Letak Terbaik AB20



2.15a Data Tata Letak Terbaik SC30

Panjang	Lebar	Koordinat X	Koordinat Y
1.52	1.97	7.07	2.17
1.80	2.22	0.92	2.59
3.45	1.16	10.27	2.89
4.49	3.56	4.07	1.78
3.33	1.20	9.85	0.89
2.18	2.29	1.09	9.40
2.14	0.93	2.33	12.42
2.14	1.40	2.33	11.25
1.83	2.73	0.92	5.06
2.39	2.51	4.71	13.11
2.55	0.78	9.06	6.68
4.15	5.78	9.85	9.96
1.86	2.69	6.84	13.19
3.83	0.78	5.87	6.68
5.59	1.97	4.98	8.13
1.67	3.58	3.12	5.35
0.93	2.15	8.23	13.92
5.59	1.43	4.98	9.83
2.18	1.83	1.09	7.34
3.84	1.30	5.85	11.20
1.26	3.17	0.63	12.13
1.06	2.82	9.70	4.88
2.23	0.45	7.43	3.38
3.68	0.82	9.67	1.90
0.45	2.24	2.06	4.94
2.59	1.54	10.00	13.62
1.66	3.60	11.17	5.27
0.49	2.05	11.76	1.28
5.21	2.68	6.56	4.94
3.26	1.23	1.89	14.33

2.15b Desain Tata Letak Terbaik SC30



2.15a Data Tata Letak Terbaik SC35

Panjang	Lebar	Koordinat X	Koordinat Y
2.66	1.13	10.94	14.33
1.91	2.62	10.41	11.38
1.11	3.59	3.48	5.77
3.56	3.93	2.04	1.97
2.94	1.36	2.26	9.12
2.61	1.91	2.91	10.80
2.83	0.71	7.96	13.94
1.97	1.52	7.96	10.83
2.77	1.81	5.60	10.74
3.53	1.70	6.57	15.14
1.33	1.51	12.53	10.65
3.17	1.89	5.31	8.89
2.16	2.32	5.57	5.25
3.46	0.87	1.99	8.00
5.02	2.59	9.16	5.25
4.90	1.23	6.31	0.61
1.05	1.91	1.08	10.80
4.52	2.21	9.16	7.65
2.07	1.94	13.31	14.73
1.90	2.63	1.97	6.26
1.37	2.92	9.77	2.50
2.95	1.02	10.24	0.52
0.50	2.00	14.75	3.38
1.25	2.39	11.08	2.76
0.63	1.58	8.77	3.17
2.60	1.54	5.35	7.18
4.60	1.30	9.37	9.41
0.50	2.00	11.62	11.06
3.27	5.51	13.30	7.14
3.72	1.08	11.32	13.23
4.65	1.94	7.13	12.61
4.24	3.31	2.69	13.41
2.79	3.59	13.11	2.59
1.47	2.73	7.72	2.59
2.57	1.17	13.59	11.99

2.15b Desain Tata Letak Terbaik SC35