PENGANTAR OPTIMASI
DALAM REKAYASA TRANSPORTASI

Febri Zukhruf
Russ Bona Frazila

DILENGKAPI VIDEO & PERANGKAT LUNAK*

ITB Press

PENGANTAR OPTIMASI

DALAM REKAYASA TRANSPORTASI

PENGANTAR OPTIMASI

DALAM REKAYASA TRANSPORTASI

Febri Zukhruf

Russ Bona Frazila

ITB Press

Hak cipta © pada penulis dan dilindungi Undang-Undang

Hak penerbitan pada ITB Press

Dilarang memperbanyak sebagian atau seluruh bagian dari buku ini tanpa izin dari penerbit.

Pengantar Optimasi dalam Rekayasa Transportasi

Penulis : Febri Zukhruf
Russ Bona Frazila
Editor : Edi Warsidi
Cetakan I : 2021
ISBN : 978-623-297-155-4

Gedung Perpustakaan Pusat ITB
Lantai Basement, Jl. Ganesa No. 10
Bandung 40132, Jawa Barat
Telp. 022 2504257/022 2534155
e-mail: office@itbpress.itb.ac.id
web: www.itbpress.itb.ac.id
Anggota Ikapi No. 034/JBA/92
APPTI No. 005.062.1.10.2018

Prakata

Dengan mengucapkan segala puji bagi Allah Swt. yang mengenggam seluruh jiwa yang ada di alam semesta, penulis bersyukur yang mendalam atas diselesaikannya buku ini.

Buku ini pada dasarnya hanya sebuah kumpulan kecil dari luasnya ilmu pengetahuan di bidang optimasi yang berserak di jagat semesta. Bagian-bagian yang berserak tersebut kemudian berusaha dikontekstualisasi ke dalam permasalahan-permasalahan sederhana pada disiplin keilmuan rekayasa transportasi. Proses ini menjadi penting seiring semakin tidak dapat dipisahkannya penerapan prinsip optimasi dalam pengambilan keputusan di bidang rekayasa transportasi. Oleh sebab itu, buku ini diharapkan menjadi salah satu pengantar untuk melakukan eksplorasi secara luas terhadap prinsip dan teknik optimasi dalam pengambilan keputusan di bidang rekayasa transportasi.

Sebagai pengantar dalam memahami optimasi, buku ini membahas fitur utama dari optimasi yang dibahas secara iteratif pada banyak kasus terkait rekayasa transportasi. Metode penyelesaian permasalahan optimasi pun dijelaskan secara runtut dan aplikatif, disertai dengan contoh dan penggunaan alat bantu perhitungan. Proses ini diharapkan dapat menjadi sarana untuk mengantarkan transformasi pembaca dari kemampuan memahami sebuah konsep menjadi sebuah kemampuan dalam penyelesaian masalah. Video penjelasan pada materi-materi utama pun diberikan dalam buku ini, sebagai bagian dari usaha memberikan pemahaman lebih baik kepada pembaca. Buku ini sangat cocok dibaca oleh mahasiswa maupun praktisi yang ingin memahami secara konseptual maupun teknikal penerapan optimasi dalam rekayasa transportasi.

Penulisan buku ini pasti tidak bisa dipisahkan dari histori pendidikan, pengajaran, penelitian dan pengabdian masyarakat yang dilakukan oleh penulis. Sehingga pada kesempatan ini pula, penulis ingin menyampaikan banyak terima kasih kepada pada para guru yang mengenalkan serta mengajarkan prinsip optimasi dalam beberapa kuliah (Matematika Rekayasa, Sistem Rekayasa, Analisis Rekayasa, Pemodelan Transportasi, dll.), yang penulis berperan sebagai mahasiswa maupun telah berkesempatan menjadi pengajar. Selain itu, ucapan terima kasih penulis sampaikan kepada guru dan kolega di KK Rekayasa

Transportasi ITB yang senantiasa mendukung dan menginspirasi selama perjalanan karir penulis. Secara khusus penulis sampaikan pula terima kasih kepada Mas Keza Harsono dan Mas Immanuel yang membantu proses penulisan buku ini. Buku ini penulis persembahkan pula pada keluarga yang senantiasa memberikan kepercayaan dan dukungan penuh terhadap penulis.

Terakhir penulis sadar bahwa buku ini tidak luput dari kesalahan dan kekurangan, sehingga penulis sangat mengharapkan masukan dan kritik untuk membangun buku ini menjadi lebih baik. Semoga buku ini memberikan manfaat kepada kebaikan dan kemajuan dunia transportasi di Indonesia.

Bandung, Agustus 2020

Penulis

Daftar Isi

Prakata ..... **v**

Daftar Isi ..... **vii**

Daftar Gambar ..... **xi**

Daftar Tabel ..... **xv**

1. Pengantar Optimasi **1**
1.2. Pendahuluan 1
1.2. Model Optimasi 3
1.3. Tipe Permasalahan Optimasi 5
1.4. Kompleksitas Masalah Optimasi 10
1.5. Teknik penyelesaian permasalahan optimasi 12

2. *Linear Programming* dengan Metode *Graph* **15**
2.1. Metode *Graph* dengan *Iso Line Profit* 15
2.2. *Metode Graph* dengan *Corner Point* 19
2.3. *Sensitivity Analysis* dengan Metode *Graph* 21

3. *Linear programming* dengan Metode *Simplex* **33**
3.1. Pengantar Metode *Simplex* 33
3.2. Konsep Dasar Perhitungan *Simplex* 34
3.3. *Simplex* Satu Fase 37
3.4 *Simplex* Dua Fase 41

4. *Post Analysis Linear programming* **49**
4.1. Pengantar *Post Analysis* 49
4.2. Perubahan Koefisien pada Fungsi Objektif 49
4.3. Shadow Prices dan Reduced Cost 56
4.4. Dual Model 61
4.5. Post Analysis dengan Ms. Excel 68

5. *Transportation Problem* **75**
5.1. Pengantar 75
5.2. *Balanced Transportation Problem* 75
5.2.1. Solusi awal dengan metode *North West* 80
5.2.2. Solusi awal dengan metode *Least Cost* 81
5.2.3. Solusi awal dengan metode *Vogel Approximation Method* (VAM) 82
5.2.4. Penentuan Variabel Basis dan Non-Basis 85
4.2.5. Evaluasi Kondisi Optimalitas 89
5.3. *Unbalanced Transportation Problem* 92

6. *Assignment Problem* **95**
6.1. Pengantar 95
6.2. Hungarian Method 96

7. Optimasi dengan *Metode Heuristic* **103**
7.1. *Greedy Heuristics* 103
7.2. *Local Search* 108

8. Optimasi dengan Metode *Metahueristik* **111**
8.1. Pengantar *Metahueristik* 111
8.2. Karakteristik Masalah yang Sesuai dengan *Metahueristik* 112
8.2.1 *Travelling Salesman Problem* 113
8.2.2. *Knapsack Problem* 114
8.2.3. *Vehicle Routing Problem* 114
8.3. Perencanaan Transportasi menggunakan *Metaheuristik* 115

9. *Particle Swarm Optimisation* (PSO) **119**
9.1. Pendahuluan 119
9.2. Tipe *Binary* PSO 120
9.2.1. *Discrete Binary* PSO (DBPSO) 120
9.2.2. *Modified Binary* PSO (MBPSO) 122
9.2.3. *Probability Discrete Binary* PSO (PBPSO) 124
9.2.1 *Modified* PBPSO (MPBPSO) 126
9.3. Aplikasi PSO dalam *Knapsack Problem* 128
9.3.1. Contoh *Knapsack* 128
9.3.2. Analisis Hasil dan Pengaturan Parameter PSO 129
9.4. Aplikasi PSO dengan *Ms. Excel* 131

10. *Genetic Algorithm* (GA) **139**
10.1. Pendahuluan 139
10.2. Kerangka dasar Genetic Algorithm 139
9.2.1. *Operator Selection* 141
9.2.2. *Operator Crossover* 142
9.2.3. Operator Mutasi 143
10.2. Simple Genetic Algorithm (SGA) 144
10.3. Aplikasi GA dengan *Ms. Excel* 145

11. *Glowworm Swarm Optimization* (GSO) **153**
11. 1. Prosedur Glowworm Swarm Optimization 153
11. 2. Contoh Aplikasi GSO 155
11. 3. Perbandingan Performa GSO dan PSO 156

12. Metode Penyelesaian Masalah Optimasi Lainnya **159**
12.1. Full enumerasi 159
12.2. Djikstra 160
12.2. Lagrange Relaxation 164

13. Optimasi Pengembangan Jaringan Transportasi dengan OPTANT PL **173**
13.1. Studi Kasus Pengembangan Jaringan Transportasi 173
13.2. Tahapan Implementasi dengan OPTANT PL 176
13.2.1. Database Pengembangan Jaringan Transportasi 176
13.2.2. Input Database Pengembangan Jaringan Transportasi 176

13.2.3. Input Parameter Metode Penyelesaian Masalah Optimasi 177
13.2.4. Penyelesaian Permasalahan Optimasi 178
13.2.5. Luaran OPTANT PL 179
13.3. Hasil Optimasi Studi Kasus 180

Referensi 183

Daftar Gambar

Gambar 1.1 Contoh pengambilan keputusan terkait rute. 1
Gambar 1.2 Ilustrasi lokal optimal dan global optimal. 11
Gambar 1.3 Ilustrasi global optimal dan beberapa lokal optimal. 11
Gambar 2.1 Fungsi kendala terkait fungsi luas. 17
Gambar 2.2 Fungsi kendala non-negatif. 17
Gambar 2.3 Ilustrasi area solusi yang mungkin (*feasible region*). 18
Gambar 2.4 Ilustrasi solusi awal dengan garis kontur. 18
Gambar 2.5 Ilustrasi proses pergeseran garis kontur awal. 19
Gambar 2.6 Ilustrasi titik ekstrim. 20
Gambar 2.7 Ilustrasi kendala mengikat (*binding constraint*). 22
Gambar 2.8 Solusi dari *corner point* method. 24
Gambar 2.9 Ilustrasi *range of optimality.* 25
Gambar 2.10 Ilustrasi kasus ketersediaan material 1 berubah menjadi 15. 26
Gambar 2.11 Ilustrasi kasus ketersediaan material 1 berubah menjadi 30. 27
Gambar 2.12 Ilustrasi kasus ketersediaan material 2 berubah menjadi 4. 29
Gambar 2.13 Ilustrasi kasus ketersediaan material 2 berubah menjadi 10. 29
Gambar 4.1 Ilustrasi lokasi *shadow price* dan *reduce cost* dari iterasi terakhir table *simplex.* 56
Gambar 4.2 Hasil *sensitivity analysis* dari *solver* untuk *shadow price.* 59
Gambar 4.3 Hasil *sensitivity analysis* dari *solver* untuk *reduced cost.* 60
Gambar 4.4 Opsi *add-ins* pada jendela *Excel options.* 68
Gambar 4.5 Opsi *solver add-in* pada jendela *add-ins.* 68
Gambar 4.6 Lokasi fitur *solver* pada *Microsoft Excel.* 69
Gambar 4.7 Format penulisan fungsi pada *Microsoft Excel.* 70
Gambar 4.8 Contoh penjabaran rumus. 70
Gambar 4.9 Input solver parameters. 71
Gambar 4.10 Input Add Constraint. 71
Gambar 4.11 Input solver results. 72
Gambar 4.12 *Answer report* dari fitur *solver.* 73
Gambar 4.13 Sensitivity report dari fitur solver. 74
Gambar 5.1 Ilustrasi Transportation Problem. 75
Gambar 5.2 Ilustrasi *Balanced Transportation Problem* (Taha, 2007). 77
Gambar 5.3 Format penyelesaian *transportation problem* (Taha, 2007). 77
Gambar 5.4 Penyusunan permasalahan. 79
Gambar 5.5 *Initial solution* pada metode *North West.* 80
Gambar 5.6 *Initial solution* pada metode *Least Cost.* 81
Gambar 5.7 Identifikasi nilai penalti pada setiap baris dan kolom. 83
Gambar 5.8 Identifikasi baris atau kolom dengan penalti terbesar. 83
Gambar 5.9 Identifikasi penalti terbesar setelah kolom lokasi-4 ditutup. 84
Gambar 5.10 Identifikasi penalti terbesar setelah kolom lokasi-2 ditutup. 84

Gambar 5.11 Identifikasi *penalty* terbesar setelah kolom ketiga ditutup. 85
Gambar 5.12 Matriks asal tujuan beserta nilai *z.* 87
Gambar 5.13 *Reduction process* dengan nilai $\boldsymbol{\beta}$ tahap 1. 88
Gambar 5.14 Nilai matriks dengan $\beta = 10$ tahap 1. 89
Gambar 5.15 *Reduction process* dengan nilai $\boldsymbol{\beta}$ tahap 2. 90
Gambar 5.16 Nilai matriks pada tahap akhir perhitungan soal 5.1. 91
Gambar 5.17 Contoh penerapan *dummy nodes* pada *unbalance transportation problem* dengan kebutuhan lebih besar daripada suplai tersedia. 93
Gambar 5.18 Contoh penerapan *dummy nodes* pada *unbalance transportation problem* dengan kebutuhan lebih kecil daripada suplai tersedia. 93
Gambar 6.1 Bentuk matriks untuk *Assignment Problem* (Taha, 2017) 95
Gambar 6.2 Penambahan garis untuk menutup nilai 0. 100
Gambar 6.3 Identifikasi *uncovered entries.* 100
Gambar 6.4 Pengurangan ke setiap *uncovered entry* dan penjumlahan ke *intersection.* 100
Gambar 6.5 Penambahan garis untuk menutup nilai 0 pada iterasi akhir. 101
Gambar 7.1 Jaringan jalan sederhana. 104
Gambar 8.1 Ilustrasi *Knapsack Problem.* 114
Gambar 9.1 Mekanisme pembaharuan vektor kecepatan PSO. 119
Gambar 9.2 Contoh pengaturan parameter MPBPSO 130
Gambar 9.3 Perbandingan performa dari algortima PSO. 131
Gambar 9.4 Contoh permasalahan optimasi pengakutan barang dengan kapasitas angkut terbatas. 132
Gambar 9.5 Rumus pada *Ms. Excel* untuk membangkitkan bilangan acak. 134
Gambar 9.6 Nilai terbaik dari pergerakan pertama di iterasi 1. 136
Gambar 10.1 Ilustrasi terkait istilah penyusun *Genetic Algorithm.* 140
Gambar 10.2 Interaksi tiga operator utama dalam GA. 141
Gambar 10.3 Contoh roda rolet. 142
Gambar 10.4 Proses *single crossover.* 143
Gambar 10.5 Ilustrasi *crossover.* 143
Gambar 10.6 Terjadinya mutasi. 144
Gambar 10.7 Rumus bilangan acak pada *Ms. Excel.* 146
Gambar 10.8 Hasil dari *copy+paste* rumus bilangan acak. 146
Gambar 10.9 Evaluasi *fitness value* GA. 147
Gambar 10.10 Rumus di *Ms. Excel* untuk menentukan terjadi kawin silang 147
Gambar 10.11 Rumus di *Ms. Excel* untuk menentukan *parent* dari *crossover.* 148
Gambar 10.12 *Parent* yang terpilih. 148
Gambar 10.13 Penentuan lokasi persilangan. 148
Gambar 10.14 Dua individu baru hasil *crossover.* 148
Gambar 10.15 Rumus di *Ms. Excel* penentuan mutasi. 149
Gambar 10.16 Contoh hasil individu bermutasi. 150
Gambar 10.17 Rumus di *Ms. Excel* untuk seleksi individu. 151
Gambar 11.1 Proses di CT yang dikonsiderasikan (Zukhruf et al, 2020). 155
Gambar 12.1 Contoh jaringan sederhana untuk kasus pencarian rute terbaik. 161

Gambar 12.2 Template label. 161
Gambar 12.3 Pembaharuan *working value Node* terdekat..... 161
Gambar 12.4 Pembaharuan Label *Node* terdekat. 162
Gambar 12.5 Pembaharuan Label *Node* pada tahap kedua. 162
Gambar 12.6 Pembaharuan Label *Node* pada tahap ketiga. 162
Gambar 12.7 Pembaharuan Label *Node* pada tahap empat..... 163
Gambar 12.8 Pembaharuan Label *Node* pada tahap lima..... 163
Gambar 12.9 Hasil akhir *Djikstra* 164
Gambar 12.10 Ilustrasi upper bounds and lower bounds permasalahan optimasi. 165
Gambar 13.1 Jaringan Jalan Studi Kasus. 174
Gambar 13.2 *Vehicle Capacity Ratio* (VCR) hasil luaran OPTANT PL. 174
Gambar 13.3 Tangkapanlayar Input Load Optimization Data..... 177
Gambar 13.4 Tangkapan layar *Input* PSO. 177
Gambar 13.5 Tangkapan layar Input PSO. 178
Gambar 13.6 Tangkapan layar penyelesaian masalah optimasi dengan PSO. 178
Gambar 13.7 Tangkapan layar keputusan optimal dengan PSO. 179
Gambar 13.8 Tangkapan layar solusi terbaik pada setiap iterasinya. 180
Gambar 13.9 Tangkapan layar solusi terbaik pada setiap iterasinya. 180

Daftar Tabel

Tabel 1.1 Waktu untuk menyelesaikan masalah dengan variasi fungsi ukuran dan kompleksitas masalah 11
Tabel 2.1 Kebutuhan dan ketersediaan material untuk produksi. 22
Tabel 3.1 Matriks untuk *simplex* iterasi-1. 40
Tabel 3.2 Matriks dengan *pivot column, pivot row*, dan *pivot number.* 40
Tabel 3.3 Matriks setelah dibagi dengan *pivot number.* 41
Tabel 3.4 Matriks dengan evaluasi *cl*. 41
Tabel 3.5 Matriks iterasi tahap 1 soal 3.2. 45
Tabel 3.6 Matriks dengan *pivot row* dan *pivot column* tahap 1 soal 3.2. 45
Tabel 3.7 Matriks iterasi tahap 2 (soal 3.2.) 46
Tabel 3.8 Matriks dengan *pivot row* dan *pivot column* tahap 2 (soal 3.2). 46
Tabel 3.9 Matriks dengan evaluasi *cl* (soal 3.2). 47
Tabel 4.1 Matriks beserta *pivot column* dan *pivot row.* 52
Tabel 4.2 Matriks hasil metode *simplex* (soal 4.1a) 52
Tabel 4.3 Input informasi pada matriks di iterasi akhir (soal 4.1b). 53
Tabel 4.4 Matriks hasil metode *simplex* (soal 4.1b) 54
Tabel 4.5 Matriks input informasi pada matriks di iterasi akhir (soal 4.1c) 55
Tabel 4.6 Matriks hasil metode *simplex* (soal 4.1c) 55
Tabel 4.7 Ilustrasi *shadow price.* 56
Tabel 4.8 Ilustrasi *reduced cost.* 57
Tabel 4.9 Matriks hasil metode *simplex* (soal 4.2a). 58
Tabel 4.10 Hasil perubahan solusi optimum pada perubahan *constraint 1.* 59
Tabel 4.11 Ilustrasi matriks akhir metode *simplex* soal 4.3(a). 65
Tabel 4.12 Matriks akhir metode *simplex* soal 4.3(c) 66
Tabel 4.13 Perubahan pada matriks *simplex.* 67
Tabel 4.14 Matriks akhir metode *simplex* soal 4.3(c) 67
Tabel 5.1 Kebutuhan setiap lokasi konstruksi. 78
Tabel 5.2 Kapasitas produksi dari *Batching Plant.* 78
Tabel 5.3 Kapasitas produksi dari setiap *Batching Plant.* 79
Tabel 5.4 Penentuan nilai *U* dan *V* tahap 1 86
Tabel 5.5 Evaluasi variabel non basis tahap 1. 87
Tabel 5.6 Rekapitulasi nilai *z* setiap variabel tahap 1. 87
Tabel 5.7 Penentuan nilai U dan V tahap 2. 89
Tabel 5.8 Evaluasi variabel non basis tahap 2. 90
Tabel 5.9 Rekapitulasi nilai *z* setiap variabel tahap 2. 90
Tabel 5.10 Rekapitulasi nilai *z* setiap variabel tahap akhir. 91
Tabel 5.11 Rekapitulasi hasil *Transportation Problem.* 92
Tabel 6.1 Biaya penawaran ke pekerjaan (dalam Milyar Rp.). 96
Tabel 6.2 Identifikasi nilai terkecil dari setiap baris. 96
Tabel 6.3 Pengurangan setiap baris dengan nilai baris terkecil. 97

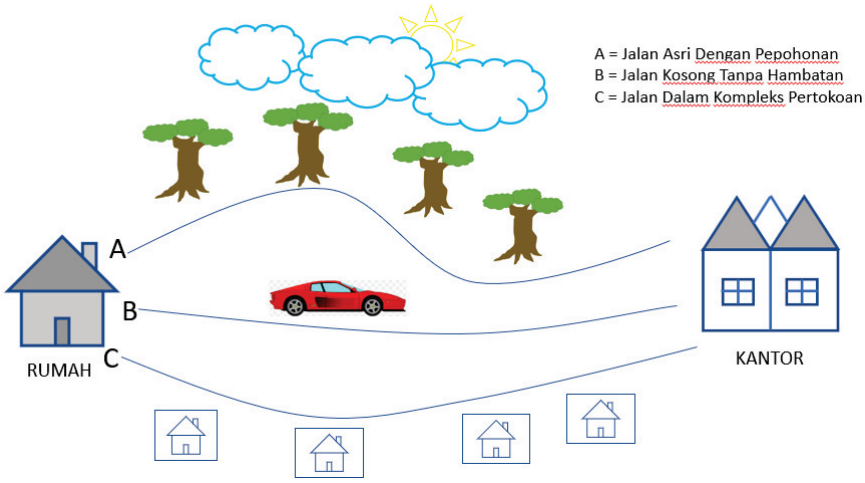
Tabel 6.4 Pengurangan setiap kolom dengan nilai kolom terkecil. 97
Tabel 6.5 Penambahan garis untuk menutupi nilai 0. 97
Tabel 6.6 Waktu perjalanan tim ke lokasi ruas yang rusak. 98
Tabel 6.7 Identifikasi baris terkecil (soal 6.2). 99
Tabel 6.8 Identifikasi kolom terkecil (soal 6.2). 99
Tabel 6.9 Hasil pengurangan setiap nilai dengan nilai terkecil baris dan kolom. 99
Tabel 7.1 Beban lalu lintas dengan nilai iri pada kondisi awal. 104
Tabel 7.2 Performa akibat pemeliharaan. 105
Tabel 7.3 Nilai fungsi tujuan pada masing-masing penanganan di setiap ruas. 106
Tabel 7.4 Nilai fungsi tujuan dan kode ruas. 106
Tabel 7.5 Urutan nilai *fitness* dan kode ruas. 107
Tabel 7.6 Penentuan urutan penanganan. 108
Tabel 7.7 Jarak antar lokasi. 109
Tabel 7.8 Nilai *fitness* solusi awal 109
Tabel 7.9 Nilai objektif solusi yang berdekatan 109
Tabel 7.10 Nilai objektif solusi yang berdekatan 110
Tabel 8.1 Contoh waktu penyelesaian permasalahan NP dan P 112
Tabel 8.2 Efek jumlah kota pada ukuran ruang pencarian TSP. 113
Tabel 9.1 Contoh permasalahan *Knapsack.* 129
Tabel 9.2 Representasi data *knapsack.* 133
Tabel 9.3 Parameter settings PSO. 133
Tabel 9.4 Tabel untuk *velocity* setiap partikel. 134
Tabel 9.5 Tabel untuk fungsi sigmoid pada kondisi awal. 134
Tabel 9.6 Tabel untuk bilangan acak. 135
Tabel 9.7 Tabel *posisi partikel.* 135
Tabel 9.8 Tabel *fitness value* dan *evaluasi fungsi kendala.* 135
Tabel 9.9 Tabel *Pbest.* 136
Tabel 9.10 Tabel pembaharuan kecepatan. 137
Tabel 10.1 Contoh % *fitness value* untuk roda rolet. 142
Tabel 10.2 *Parameter settings GA.* 145
Tabel 10.3 Hasil *fitness value* setelah pengecekan batasan kapasitas. 147
Tabel 10.4 Contoh hasil *crossover.* 147
Tabel 10.5 Berat dan *fitness value* individu baru. 148
Tabel 10.6 Contoh hasil penentuan mutasi. 149
Tabel 10.7 Contoh penggunaan roda rolet. 150
Tabel 10.8 Individu yang melanjutkan ke generasi selanjutnya. 151
Tabel 11.1 Perbandingan kinerja GSO dan PSO. 157
Tabel 12.1 Waktu tempuh antar simpang (menit). 159
Tabel 12.2 Enumerasi pilihan rute 159
Tabel 12.3 Contoh segmen jalan untuk aplikasi *lagrange relaxation.* 166
Tabel 12.4 Nilai IRI pada masing-masing keputusan 168
Tabel 12.5 Implementasi *multiplier adjustment.* 170
Tabel 13.1 Pilihan segmen untuk pelebaran jalan. 175
Tabel 13.2 Keterangan data dalam datopt 176

1. Pengantar Optimasi

1.1. Pendahuluan

Manusia dalam kehidupannya selalu dihadapkan kepada berbagai masalah yang menuntut suatu pengambilan keputusan. Secara umum sebelum mengambil sebuah keputusan, manusia biasanya berusaha mengidentifikasi terlebih dahulu permasalahan tersebut. Identifikasi dimulai dengan memahami secara seksama urutan masalah yang dihadapi, beserta kondisi pendukung ataupun pembatasnya. Setelah memahami masalah tersebut, umumnya manusia berusaha menyederhanakan permasalahan itu agar dapat diperoleh gambaran tentang berbagai alternatif keputusan yang mungkin diambil untuk menyelesaikan permasalahan tersebut. Pada proses iteratif (i.e., berulang) ini, sesungguhnya manusia secara sadar maupun tidak sadar sedang melakukan optimasi atas alternatif keputusan yang mungkin diambil.

Proses menimbang keputusan dengan memikirkan ketercapaian tujuan dan dampak dari keputusan tersebut, sejatinya merupakan proses optimasi. Sehingga, optimasi dapat diartikan sebagai proses menemukan solusi terbaik yang tersedia, dari suatu fungsi tujuan (i.e., *objective function*) yang ditetapkan, dengan tetap memperhatikan kendala (i.e. *constraint*) yang ada.



Gambar 1.1 Contoh pengambilan keputusan terkait rute.

Optimasi selalu berkaitan dengan yang apa yang disebut dengan *decision making* atau pengambilan keputusan. Dalam bidang rekayasa ataupun perencanaan transportasi,

terdapat berbagai permasalahan pengambilan keputusan yang harus dihadapi, mulai dari keputusan yang bersifat sederhana hingga keputusan yang bersifat kompleks.

Gambar 1.1 memberikan contoh pengambilan keputusan terkait rute, dimana terdapat berbagai kombinasi segmen jalan yang membentuk sebuah rute. Sehingga dibutuhkan proses pengambilan keputusan untuk memutuskan pilihan rute mana yang akan diambil, yang dapat disebut sebagai variabel keputusan. Variabel keputusan terkait rute yang diambil akan sangat terkait tujuan dari perjalanan itu sendiri. Sebagai contoh, rute yang mengkombinasikan jalan-jalan dengan arus lalu lintas rendah akan lebih menarik untuk dipilih jika tujuan dari perjalanan itu adalah menghindari kemacetan dan dapat cepat sampai ke lokasi tujuan (e.g., rute B). Sementara itu, rute dengan kombinasi jalan yang memiliki pusat pertokoan atau tempat singgah yang banyak akan menjadi pilihan jika tujuan dari perjalanan adalah memaksimalkan jumlah lokasi yang dikunjungi dalam satu perjalanan (e.g., Rute C). Meskipun pilihan ini harus berhadapan dengan kecepatan perjalanan yang rendah. Pilihan-pilihan rute tersebut dapat berubah jika fungsi tujuannya berubah pula. Permasalahan ini juga dapat menjadi semakin kompleks manakala melibatkan jumlah segmen jalan yang semakin banyak. Karena dengan adanya jumlah segmen jalan yang semakin banyak, maka jumlah pilihan rute akan semakin besar sehingga proses pengambilan keputusan akan semakin sulit pula.

Contoh di atas adalah bagian dari pengambilan keputusan yang selalu melibatkan begitu banyak alternatif pilihan dalam rangka mencapai tujuan dari pengambilan keputusan tersebut. Pilihan-pilihan yang ada akan diputuskan berdasarkan kesesuaian dengan tujuan dari pengambilan keputusan tersebut. Tujuan dari pengambilan keputusan tersebut dapat berupa maksimalisasi maupun minimalisasi suatu parameter. Sebagai contoh, pemilihan rute jalan dapat memiliki tujuan untuk meminimalisasi waktu tempuh dan jarak perjalanan atau berupa memaksimalisasi jumlah tempat yang disinggahi selama perjalanan, atau berupa memaksimalisasi kenyamanan dalam perjalanan. Keputusan yang diambil terhadap begitu banyak pilihan tersebut kemudian akan sangat ditentukan oleh tujuan dari pengambilan keputusan tersebut.

Selain variabel yang akan diputuskan berdasarkan kepada tujuan dari pengambilan keputusan, setiap pengambilan keputusan dihadapkan juga kepada batasan atau kendala yang dapat mempengaruhi keputusan yang diambil. Sebagai contoh, dalam proses pengambilan keputusan rute, terkadang kita dihadapkan kepada batasan, seperti batasan

waktu tiba dilokasi tujuan atau adanya segmen jalan yang tidak boleh dilalui yang kesemuanya akan mempengaruhi keputusan akan rute tersebut. Pada contoh lain, seperti proses pengambilan keputusan terkait moda yang akan digunakan, kondisi keuangan, kondisi cuaca, waktu yang tersedia dapat dianggap sebagai fungsi kendala yang membatasi dalam menentukan moda yang akan diambil. Pada kasus ini, anggaplah kita memiliki variabel keputusan berupa sepeda motor dan mobil pribadi, dimana fungsi tujuan kita adalah meminimalkan waktu perjalanan pada saat jam sibuk. Pada cuaca cerah, mungkin sepeda motor akan menjadi pilihan utama untuk mencapai tujuan tersebut. Meskipun pilihan tersebut dapat berubah manakala terdapat batasan terkait adanya hujan lebat, yang akan mempengaruhi keputusan.

Dengan mengambil contoh-contoh di atas, proses pengambilan keputusan kemudian dapat distrukturisasi menjadi tiga fitur utama yaitu:

- Melibatkan begitu banyak pilihan yang harus diputuskan.
- Memasukkan fungsi tujuan dari pengambilan keputusan.
- Mempertimbangkan adanya batasan atau kendala dalam pengambilan keputusan.

Ketiga fitur di atas kemudian digunakan untuk mendefinisikan arti dari optimasi. Optimasi dapat didefinisikan sebagai proses **pengambilan keputusan** dari **begitu banyak pilihan** yang tersedia dalam rangka **memaksimalkan atau meminimalkan** suatu **fungsi tujuan** akan tetapi tidak **melanggar batasan** yang ada. Oleh karenanya, keputusan dari permasalahan optimasi disebut sebagai keputusan optimal manakala:

- Mampu memaksimalkan atau meminimalkan fungsi tujuan
- Mampu tidak melanggar kendala atau batasan yang ada

1.2. Model Optimasi

Karena sangat erat kaitannya dengan permasalahan nyata manusia, permasalahan optimasi yang digambarkan melalui persamaan matematika telah berkembang sangat pesat. Perkembangan ini didorong oleh adanya beberapa tipikal utama dari permasalahan di bidang kerekayasaan, yaitu:

- Melibatkan begitu banyak alternatif keputusan beserta kombinasinya.
- Memiliki tujuan spesifik yang umumnya dirumuskan sebagai indikator kinerja.
- Umumnya dapat disederhanakan dan direpresentasikan dengan bentuk model matematika.

Permasalahan optimasi yang diformulasikan ke dalam bentuk model matematika umum dikenal sebagai *Mathematical Optimization* (i.e., optimasi matematika). Secara terminologi, optimasi dalam bentuk model matematika adalah gabungan seni dan sains dalam mencari dan menentukan solusi terbaik dari sebuah permasalahan optimasi berbentuk matematika yang memiliki fungsi tujuan dan fungsi kendala yang telah ditetapkan sebelumnya (Snyman, J.A., 2006.).

Penggunaan matematika dalam penyelesaian permasalahan optimasi memungkinkan untuk menghindari proses pengambilan keputusan secara naratif. Sebaliknya, penerapan kerangka optimasi ini memungkinkan dilakukannya proses pencarian keputusan terbaik secara terkuantifikasi dan minim bias. Penyederhanaan permasalahan ke dalam bentuk model matematis dapat pula memudahkan dalam memahami permasalahan secara lebih terperinci. Pertanyaan yang terkait pengambilan keputusan dapat dijawab secara kuantitatif, sebagai contoh, variabel keputusan mana yang dapat memaksimalkan fungsi tujuan, ataupun fungsi kendala mana yang paling berpengaruh kepada solusi optimal. Oleh karena itu, menjadi sangat penting untuk dapat mentransformasi permasalahan optimasi yang bersifat naratif menjadi bentuk persamaan matematika. Pengejawantahan permasalahan optimasi ke dalam model matematika sesungguhnya tidak terlepas dari tiga fitur utama optimasi yang disampaikan sebelumnya. Sehingga secara umum, model matematis untuk merepresentasikan permasalahan optimasi dapat dilihat sebagai berikut:

$$Min\ f(x), X = [x_1, x_2, x_3, \ldots\ldots, x_n]^T \in \mathrm{R^n} \tag{1.1}$$

Subject to

$$g_j(x) \leq 0, j = 1,2,3, \ldots\ldots, m \tag{1.2}$$

$$h_j(x) = 0, j = 1,2,3, \ldots\ldots, r \tag{1.3}$$

Dari Persamaan (1.1) hingga (1.3) di atas dapat diketahui bahwa:

- X merepresentasikan variabel keputusan yang harus diputuskan dalam proses optimasi
- $f\ (x)$ merepresentasikan fungsi tujuan yang mengandung variabel keputusan x yang harus diminimalkan (dapat pula dimaksimalkan) lihat Persamaan (1.1)
- $g\ (x)$ dan $h\ (x)$ merepresentasikan fungsi kendala yang membatasi pengambilan keputusan. Fungsi ini dapat berupa pertidaksamaan $g\ (x)$ maupun bentuk persamaan $h\ (x)$ (lihat Persamaan (1.2) dan (1.3))

Dengan mendasarkan kepada model matematika tersebut, dapat terlihat bahwa model tersebut tetap mengarah kepada tiga fitur utama permasalahan optimasi yang telah dijelaskan sebelumnya, yakni:

- Adanya variabel keputusan yang begitu banyak (i.e., kumpulan dari-x), yang harus diputuskan lihat Persamaan (1.1)
- Keputusan-x tersebut harus diputuskan dalam rangka memaksimalkan atau meminimalkan fungsi tujuan ($f(x)$ lihat Persamaan (1.1)),
- Akan tetapi juga keputusan tersebut harus memenuhi syarat dari sisi fungsi kendala ($g(x)$ dan $h(x)$ pada Persamaan (1.2) dan (1.3)).

Solusi optimal (x^*) kemudian diartikan sebagai solusi yang memberikan nilai yang paling maksimum atau minimum dari fungsi tujuan $f(x^*)$ akan tetapi tidak melanggar fungsi kendala yang ada ($g(x)$ dan $h(x)$).

1.3. Tipe Permasalahan Optimasi

Adanya aspek kemanfaatan yang besar menyebabkan model optimasi berkembang menjadi berbagai macam tipe dengan metode penyelesaiannya yang spesifik. Karena setiap jenis permasalahan optimasi akan memiliki metode penyelesaiannya yang sangat berbeda, menjadi penting untuk mengetahui tipe-tipe permasalahan optimasi. Hal ini penting tidak hanya dalam rangka mendefinisikan secara tepat permasalahannya, akan tetapi pula dalam rangka menemukan metode yang paling tepat dalam menyelesaikan tipe permasalahan optimasi tersebut. Terdapat banyak sekali tipe dan varian dari permasalahan optimasi, meskipun buku ini membagi tipe permasalahannya menjadi 5 bagian utama saja yaitu:

- *Discrete* dan *Continous;*
- *Linear* dan *Non Linear;*
- *Unconstrained* dan *Constrained ;*
- *Stochastic* dan *Deterministic;*
- *Single objective* dan *Multiobjectives.*

Discrete* dan *Continous

Pada dasarnya masalah optimasi tergantung pada jenis variabel keputusan yang ditangani. Jenis variabel keputusan secara umum dapat dibagi menjadi dua kluster utama, yaitu variabel keputusan yang berbentuk *discrete* dan variabel yang berbentuk *continuous.* Variabel keputusan dari model optimasi diskrit selalu bersifat terputus-putus, sedangkan

tipe optimasi *continuous* bersifat menerus. Sebagai ilustrasi, berikut penjelasan dari konsep variabel diskrit dan kontinu.

- *Discrete* dibentuk dengan menggunakan bilangan biner (1 atau 0) dan bilangan integer (..., -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5,...)
  - Contoh bilangan biner: permasalahan optimasi untuk menentukan apakah sebuah proyek pengembangan jalur kereta api akan dilakukan (dimodelkan dengan 1) atau tidak dilakukan (dimodelkan dengan 0) dalam jaringan transportasi.
  - Contoh bilangan integer: permasalahan optimasi untuk menentukan lokasi halte mana yang akan dikunjungi secara berurutan. (i) Apakah berawal dari Halte 1, kemudian Halte 2, lalu ke Halte 3, ataukah (ii) bergerak awal dari Halte 2 menuju Halte 1 baru ke Halte 3. Kedua variabel keputusan tersebut dapat direpresentasikan dalam bentuk bilangan integer yaitu adalah (1-2-3) untuk variabel keputusan (i) atau (2-1-3) untuk variabel keputusan (ii).
- *Continuous* dibentuk oleh bilangan real (1; 12.38; -0.8625; π (pi); 198). Contohnya adalah penentuan lebar dalam pekerjaan pelebaran jalan, apakah 3,51; 3,61; 3,72 dst.

Linear* dan *Non Linear

Jenis lain yang umumnya didefinisikan dalam permasalahan ataupun model optimasi adalah masalah optimasi yang berbentuk linear ataupun non linear. Masalah optimasi linear memiliki fungsi tujuan dan kendala yang bersifat linear, sementara kondisi sebaliknya untuk permasalahan optimasi berjenis non linear. Permasalahan optimasi linear dan non linear umum ditemui dalam bentuk *mathematical programming model* yang dianggap sukses menyelesaikan berbagai masalah optimasi.

Berdasarkan karakteristik masalahnya, *mathematical programming model* dapat dibagi menjadi dua kategori besar, yaitu *Linear programming* (LP) dan *Non-Linear programming* (NLP). Sesuai namanya, LP menunjukkan pemrograman permasalahan optimasi yang fungsi tujuannya (*objective function*) dan/atau fungsi kendalanya (*contraints*) merupakan fungsi linear (lihat Persamaan (1.4)). Metode *simplex* merupakan salah contoh metode yang umum digunakan dalam LP, dan dapat menjamin diperolehnya solusi global optimal.

$$\min f(x_1, x_2) = x_1 + x_2$$

Subject to (1.4)

$$x_1 + x_2 \leq 2$$

Sementara itu, NLP mengacu kepada pemrograman permasalahan optimasi yang fungsi tujuannya (*objective function*) dan/atau fungsi kendalanya (*contraints*) merupakan fungsi non linear (lihat Persamaan (1.5)). Karena karakteristik non linearnya, permasalahan ini sedikit lebih sulit untuk diselesaikan. Meskipun untuk permasalahan berskala kecil hingga menengah, beberapa metode eksak dapat menjamin diperolehnya global optimal dalam waktu yang bisa diterima. Pada permasalahan berskala besar, metode berbasis pendekatan dapat menjadi salah satu solusi dalam menyelesaikan NLP.

$$\min f(x_1, x_2) = {x_1}^2 + x_2$$

Subject to (1.5)

$${x_1}^2 + x_2 \leq 12$$

Constraint* dan *Unconstraint

Permasalahan optimasi dalam kerekayasaan umumnya memiliki fungsi kendala atau biasa disebut sebagai *constrained optimization*, karena tanpa adanya fungsi kendala maka keputusannya akan sangat mudah untuk diputuskan. Meskipun secara matematis fungsi kendala ini dimungkinkan untuk dirilis, sehingga secara matematis permasalahan optimasi akan berubah menjadi permasalahan optimasi tanpa fungsi kendala (*unconstrained optimization*).

Sebagai contoh, Persamaan (1.6) mengilustrasikan permasalahan dalam menentukan apakah suatu segmen jalan-l (x_l) akan dipelihara atau tidak pada suatu jaringan jalan. Permasalahan ini dapat dimodelkan dalam bentuk bilangan biner. Permasalahan optimasi ini berusaha untuk memaksimalkan nilai total selisih ketidakrataan (*international roughness index*-IRI) jika tidak dipelihara (${H_l}^0$) dan jika dipelihara ($H_l(x)$) pada jaringan jalan (Z_1). Nilai IRI akan sangat tergantung kepada aksi pemeliharaan yang dilakukan pada jalan-l ($H_l(x)$), jika dipelihara maka nilai IRI akan semakin kecil sehingga selisihnya akan semakin besar. Sementara jika tidak dipelihara maka selisihnya akan bernilai 0. Untuk memaksimalkan fungsi tujuan tersebut perlu diambil keputusan terkait jalan mana saja yang akan dipelihara.

$$Maximize\ Z_1 = \sum_{l=1}^{L} {H_l}^0 - H_l(x) \tag{1.6}$$

Permasalahan ini menjadi kompleks karena aksi pemeliharaan- x pada segmen jalan-l akan membutuhkan biaya C_l. Sementara pihak yang berwenang dalam memelihara jaringan jalan tersebut pasti memiliki batasan anggaran (B). Sehingga perlu dipikirkan secara matang segmen mana saja yang akan dipelihara karena adanya batasan dalam anggaran pemeliharaan jalan. Kondisi ini kemudian direpresentasikan pada Persamaan (1.7), yang dapat dianggap sebagai fungsi kendala pada permasalahan optimasi ini. Oleh karenanya, permasalahan optimasi ini dapat dianggap sebagai permasalahan optimasi *constrained.*

$$\sum_{l=1}^{L} C_l x_l \leq B \tag{1.7}$$

Meskipun secara matematis fungsi kendala ini dapat dihilangkan. Banyak strategi untuk meliris *constraint* ini, salah satunya adalah memasukkan fungsi kendala ke dalam fungsi tujuan.

$$Maximize\ Z_1 = \frac{\sum_{l=1}^{L} {H_l}^0 - H_l(x)}{\sum_{l=1}^{L} C_l x_l} \tag{1.8}$$

$$Maximize\ Z_1 = \left(\sum_{l=1}^{L} {H_l}^0 - H_l(x)\right) - \varepsilon\left(B - \sum_{l=1}^{L} C_l x_l\right) \tag{1.9}$$

Persamaan (1.8) menunjukan salah satu cara memasukkan fungsi kendala ke dalam fungsi tujuan, dimana aspek biaya dijadikan pembagi. Persamaan ini akan selalu memprioritaskan segmen-segmen jalan dengan biaya rendah tapi mampu menghasilkan selisih IRI yang paling besar. Meskipun proses ini masih memungkinkan bahwa solusi yang dihasilkan akan tetap melebihi anggaran yang ada. Persamaan (1.9) memberikan alternatif lain dalam melepaskan fungsi kendala, dimana selisih antara *budget* tersedia dengan biaya yang dibutuhkan sebagai faktor untuk mengurangi nilai selisih IRI. Permasalahan optimasi dengan tidak adanya fungsi kendala dapat disebut permasalahan *unconstrained*.

Stochastic* dan *Deterministic

Tipe lain yang muncul dalam kerangka optimasi adalah adanya permasalahan yang konsiderasi fenomena keacakan. Sehingga permasalahan optimasi yang mempertimbangkan aspek ketidakpastian (*uncertainty*), probabilitas kejadian (*probabilistic*) atau keacakan (*randomness*) dari suatu fenomena umum disebut sebagai permasalahan optimasi stokastik. Sementara itu dilain pihak, permasalahan optimasi

deterministik tidak mempertimbangkan aspek keacakan didalam prosesnya. Sebagai contoh, anggaplah kita berada dalam permasalahan optimasi untuk menentukan rute yang memberikan waktu perjalanan tercepat, sehingga fungsi tujuan dalam masalah ini adalah meminimalkan waktu perjalanan. Waktu perjalanan akan sangat bergantung kepada arus lalu lintas (μ) di jalan-x yang membentuk rute.

$$Minimize\ t(x, \mu) \tag{1.10}$$

Pada pendekatan deterministik kita menganggap bahwa arus lalu selalu tetap seperti yang tergambarkan pada Persamaan (1.10). Meskipun kita ketahui bahwa arus lalu lintas memiliki variasi selama rentang waktu. Pendekatan stokastik memiliki karakteristik untuk mempertimbangkan variasi ini, umumnya dengan menganggap bahwa arus bervariasi mengikuti suatu pola distribusi tertentu.

$$Minimize\ E_{r\in\Omega}[t(x, \mu(r))]) \tag{1.11}$$

Bilangan acak umumnya disertakan didalam model kemudian metode ini umumnya memasukkan bilangan acak (i.e., r) untuk menggambarkan kejadian yang bersifat tidak pasti (i.e., Ω) atau keacakan dari arus lalu lintas tersebut. Persamaan (1.11) memberikan ilustrasi fungsi tujuan pada kasus optimasi yang sama, akan tetapi mempertimbangkan keacakan dari arus lalu lintas. Fungsi tujuan tersebut menganggap bahwa arus lalu lintas akan dibangkitkan oleh bilangan acak dengan mengikuti sebuah distribusi tertentu. Secara teknis, bilangan acak ini akan dibangkitkan dalam beberapa kali percobaan (umumnya lebih dari 500), sehingga nilai dari waktu tempuh akan bervariasi dan dapat membentuk distribusi nilai tertentu. Kondisi ini berbeda dengan pendekatan deterministik yang memungkinkan hanya ada satu nilai dalam penyelesaian di fungsi tujuannya.

Selain itu, penting untuk dicatat, bahwa nilai *fitness* dari model optimasi stokastik umumnya disebut sebagai *expected value*, yang merepresentasikan nilai yang diekspektasikan akan muncul meskipun adanya variasi atau keacakan dari arus lalu lintas ini. Nilai ekspektasi ini biasanya diperoleh dengan menggunakan pendekatan persentil dari distribusi nilai *fitness* hasil perulangan. Sebagai contoh, dari hasil percobaan 500 kali diperoleh bahwa nilai *fitness* (i.e., waktu tempuh) dengan persentil 90% adalah 50 menit. Kondisi ini mengambarkan bahwa nilai *fitness* optimum (dan yang lebih kecil dari 50 menit) diekspektasikan akan muncul dalam 90% kejadian.

Single objective* dan *Multi objectives

Tipe permasalahan optimasi lain yang muncul adalah tipe optimasi yang memiliki satu atau beberapa fungsi tujuan. Permasalahan optimasi yang hanya memiliki satu tujuan dikenal sebagai *single objective optimization problem*. Beberapa ilustrasi sebelumnya menunjukkan permasalahan optimasi dengan satu fungsi tujuan. Sementara jika permasalahan optimasi memiliki beberapa fungsi tujuan disebut *multi objectives optimization problem*. Permasalahan *multi objectives* umumnya memiliki beberapa fungsi tujuan yang saling bertolak belakang, yang artinya meningkatkan nilai pada fungsi tujuan pertama akan menyebabkan penurunan nilai *fitness* dari fungsi tujuan yang lainnya.

Sebagai contoh, mari kita tinjau kasus pemeliharaan jalan pada Persamaan (1.12) dan Persamaan (1.13). Pemerintah berharap dapat meminimalkan biaya pemeliharaan (i.e., $C_a\ (k,p)$), di sisi lain pengguna jalan berharap biaya operasi kendaraan akan turun dengan baiknya kondisi permukaan jalan (i.e., $C_u(v,p)$). Peningkatan jumlah jalan yang dipelihara (p) akan meningkatkan biaya investasi pemeliharaan dari sisi pemerintah C_a. Akan tetapi pada kondisi sebaliknya, usaha ini kemudian akan menurunkan biaya operasi kendaraan pengguna jalan (C_u). Kondisi yang sebaliknya dapat terjadi, penurunan biaya pemeliharaan akan menyebabkan kenaikan biaya operasi kendaraan pengguna karena adanya jalan yang tidak dipelihara. Adanya dua fungsi tujuan yang saling ber 'konflik' satu sama lain menyebabkan permasalahan optimasi ini dapat dianggap sebagai permasalahan optimasi dengan fungsi tujuan jamak.

$$Minimize\ C_a(k,p) \tag{1.12}$$

$$Minimize\ C_u(v,p) \tag{1.13}$$

1.4. Kompleksitas Masalah Optimasi

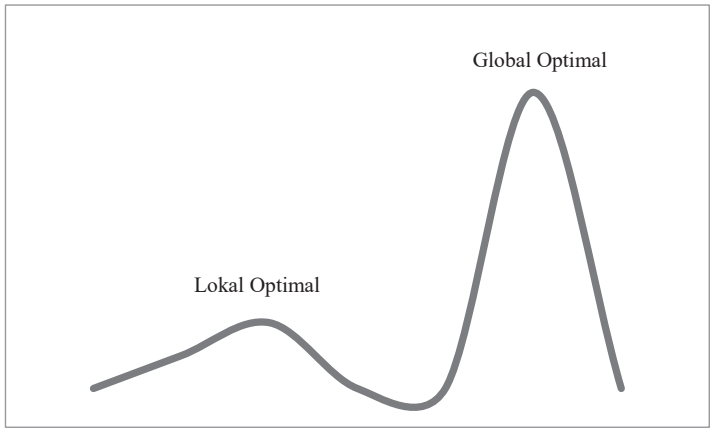
Kompleksitas masalah merupakan suatu istilah yang umum kita dengar dalam kehidupan sehari-hari. Istilah ini biasanya mengambarkan kerumitan masalah yang menuntut implementasi solusi yang terkadang tidak kalah sulit. Sesuai dengan karakteristik model matematika yang berusaha menggambarkan permasalahan manusia, dalam model optimasi juga dikenal istilah kompleksitas masalah. Meskipun dalam model optimasi, istilah ini biasanya menunjukkan kerumitan masalah yang memaksa algoritma atau metode pemecahan masalah untuk mengalokasikan waktu lebih banyak, maupun mencari solusi dalam ruang masalah yang lebih besar atau lebih sulit. Oleh karena itu, kompleksitas

masalah umumnya ditentukan oleh ukuran masalah dan waktu penyelesaiannya. Tabel 1.1 mengilustrasikan kerumitan masalah yang berkorelasi positif dengan kebutuhan waktu penyelesaian masalah. Tabel tersebut memberikan ilustrasi bahwa masalah yang dikategorikan kompleks, sangat sensitif pada kenaikan ukuran masalah, yang selanjutnya berdampak pada kenaikan secara signifikan kepada kebutuhan waktu penyelesaian masalah.

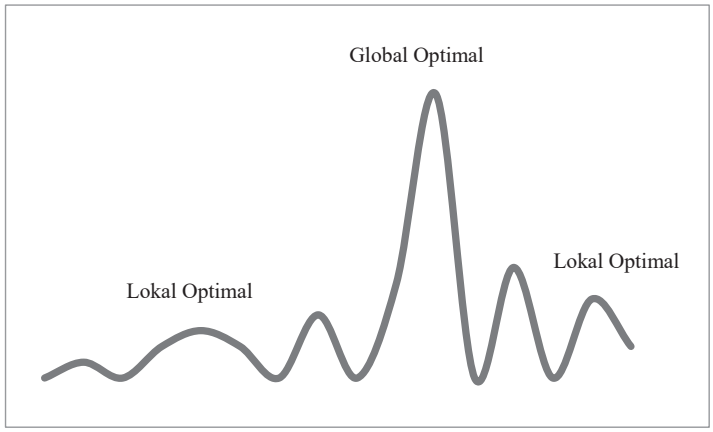
Tabel 1.1 Waktu untuk menyelesaikan masalah dengan variasi fungsi ukuran dan kompleksitas masalah.

Kompleksitas Masalah	**Ukuran Masalah**				
	10	**20**	**30**	**40**	**50**
O(x)	0.00001 s	0.00002 s	0.00003 s	0.00004 s	0.00005 s
$O(x^2)$	0.0001 s	0.0004 s	0.0009s	0.0016 s	0.0025 s
$O(x^5)$	0.1 s	0.32 s	24.3 s	1.7 menit	5.2 menit
$O(2^x)$	0.001 s	1.0 s	17.9 menit	12.7 hari	35.7 hari
$O(3^x)$	0.059 s	58.0 menit	6.5 tahun	3855 abad	2×10^8 abad

Sumber: (Garrey and Jhonson, 1979).



Gambar 1.2 Ilustrasi lokal optimal dan global optimal.



Gambar 1.3 Ilustrasi global optimal dan beberapa lokal optimal.

Selain ukuran masalah dan waktu penyelesaiannya, hal lain yang perlu diperhatikan dalam model optimasi adalah adanya solusi lokal optimal dan solusi global optimal. Sebuah solusi dianggap sebagai solusi global optimal manakala solusi yang dihasilkan memiliki nilai yang lebih baik daripada semua solusi yang tersedia. Sementara, solusi lokal optimal dapat diartikan sebagai solusi yang memiliki nilai lebih baik daripada solusi yang ada di sekitarnya, meskipun solusi ini tidak lebih baik daripada solusi global optimal. Gambar 1.2 dan Gambar 1.3 menunjukkan ilustrasi dari kedua definisi tersebut. Selain mengambarkan perbedaan solusi lokal optimal dan solusi global optimal, Gambar 1.2 secara spesifik mengilustrasikan masalah dengan banyak solusi lokal optimal. Jenis masalah dengan karakter seperti ini termasuk dalam kategori masalah yang sulit dipecahkan, karena algoritma penghasil solusi berpotensi besar dapat terjebak dalam solusi lokal optimal.

1.5. Teknik penyelesaian permasalahan optimasi

Terdapat 2 metode menyelesaikan optimasi yaitu:

- *Exact-based Method*

  Contoh: *Linear programming, Integer Programming, Non Linear programming, Dynamic Programming, Branch and Bound.*
- *Approximation-based Method*

Secara umum, metode pemecahan masalah optimasi dapat dibagi menjadi dua kategori besar, yaitu metode eksak (*exact method*) dan metode pendekatan (*approximate method*). Metode eksak memiliki keunggulan dalam jaminan terhadap hasil, dimana metode ini menjamin adanya solusi yang eksak serta global optimal. Metode *the branch-and-bound*, *branch-and-cut method* merupakan contoh dari metode berbasis eksak yang umum digunakan untuk menyelesaikan permasalahan optimasi. *The branch-and-bound* merupakan metode yang pertama kali dikembangkan oleh Land dan Doig pada tahun 1960. Metode ini telah digunakan untuk menyelesaikan berbagai masalah optimasi, terutama masalah optimasi kombinatorial. Metode ini juga dianggap efektif untuk menyelesaikan masalah yang berjenis *mixed integer linear programs* (MILP). Metode ini melakukan enumerasi sempurna (*complete enumeration*) secara implisit. Penggunaan batas dari fungsi yang dioptimasi yang dikombinasikan dengan solusi terbaik yang dihasilkan, membuat metode ini dapat melakukan enumerasi sempurna secara implisit (Clausen, 1999). Ide dasar dari metode ini adalah membagi masalah menjadi sebuah urutan sub-masalah, dan menerapkan enumerasi yang sistematis untuk semua kandidat dengan menggunakan batas

atas dan batas bawah dari fungsi yang sedang dioptimalkan. *The branch-and-cut* (BC) merupakan pengembangan lanjut dari metode BB dengan mengabungkan metode *cutting plane* dalam algoritma BB. Penjelasan lebih lanjut tentang metode BC dapat dilihat pada Mitchell (1999) dan referensinya. Selain kedua algoritma tersebut (i.e., BB dan BC) terdapat beberapa metode lain untuk menemukan solusi yang eksak, ikhtisar dari berbagai metode eksak dapat dilihat pada Laporte (1992) dan referensi di dalamnya.

Meskipun metode eksak dapat menjamin diperolehnya solusi yang eksak dan global optimal, metode ini memiliki kelemahan dalam waktu penyelesaian yang relatif lebih lama dibandingkan *approximate method*. Oleh karena itu, derajat kompleksitas masalah terkadang memaksa peneliti ataupun praktisi untuk memilih *approximate method,* meskipun tidak menjamin diperolehnya solusi yang eksak dan optimal global. *Approximate-based method* merupakan metode yang hasilnya bersifat pendekatan dan tidak bisa menjamin hasil yang global optimal. Akan tetapi tidak menutup kemungkinan bahwa metode ini bisa mencapai hasil yang global optimal. Dalam kerangka permasalahan optimasi pada rekayasa transportasi, metode berbasis pendekatan umum digunakan, seperti metode yang berlandaskan *heuristik* dan *metahueristik*.

Pada karakter masalah yang relatif kompleks, metode berbasis *heuristik* dapat digunakan sebagai alat bantu pemecahan masalah. Meskipun tidak mampu menjamin untuk menghasilkan solusi yang global optimal, metode berbasis *heuristik* di klaim mampu menghasilkan solusi yang rasional dan praktikal dalam waktu yang lebih singkat.

Heuristik dapat diartikan sebagai sebuah teknik yang terdiri dari sebuah atau sekelompok aturan atau kaidah yang berusaha mencari atau menemukan solusi yang baik dalam waktu yang dapat diterima. Teknik ini menjalankan kaidah setahap demi setahap, secara berurutan dan menghasilkan solusi berdasarkan informasi yang diperoleh selama proses berlangsung (Shapiro, 2001; Breedam, 2001). Metode ini akan menghentikan proses eksekusinya, manakala tidak ditemukannya peningkatan solusi dari tahap-tahap yang telah dilalui, atau dengan kata lain, solusi yang dihasilkan sama dengan solusi yang telah ditemukan sebelumnya. Atas dasar karakteristik tersebut, *heuristik* sering dianggap sebagai metode optimasi lokal (*local optimisation approach*).

Karakteristik pencarian yang bersifat lokal, membuat metode *heuristik* berpotensi terjebak pada solusi lokal optimal. Oleh karena itu, para peneliti berusaha mengembangkan metode lain yang mampu menghindari jebakan solusi lokal optimal, yang dikenal sebagai

metaheuristik. Penambahan kata "*meta*", berkesesuaian dengan istilah aslinya dalam bahasa Yunani, yang berarti metodologi di level yang lebih tinggi atau lebih atas. Sehingga *metaheuristik* dapat didefinisikan sebagai sebuah proses iteratif yang memandu (dan/atau memodifikasi) sub-ordinat *heuristik* untuk menghasilkan solusi yang mendekati optimal secara efisien. Metode ini memadukan strategi di level yang lebih tinggi dengan *heuristik* yang berbasis peningkatan solusi lokal (Glover and Kochenberger, 2003; Breedam, 2001). Proses iteratif ini dilaporkan mampu keluar dari jebakan solusi lokal optimal, dan sering dianggap sebagai bagian dari metode kecerdasan buatan untuk masalah optimasi (Coppin, 2004). Dalam kerangka bidang transportasi dan logistik, metode *metaheuristik* telah diaplikasikan pada beragam permasalahan dengan karakteristik yang bervariasi (lihat Griffis et al., 2012).

Pembaca yang tertarik untuk melihat penjelasan dalam bentuk audio-visual terkait Bab ini dapat mengunjungi playlist youtube berikut:

https://bit.ly/Playlist-PengantarOptimasi

2. *Linear Programming* dengan Metode *Graph*

Linear programming berkembang dengan pesat sejak perang dunia ke-2, ketika dibutuhkan suatu sistem untuk memaksimalkan efisiensi sumber daya yang ada demi memenangkan peperangan (Lewis, 2008). Pada perang dunia ke-2, bidang ilmu terkait *linear programming* digunakan untuk menentukan tinggi pesawat yang paling optimal dalam rangka menghasilkan jumlah kehancuran kapal yang semakin banyak di bawahnya. Semakin tinggi atau jauh kapal terbang, maka tingkat kehancuran akan semakin sedikit, namun jika terlalu dekat, kemungkinan pesawat tertembak kapal laut lainnya juga semakin besar. Oleh karenanya, diperlukan alat bantu optimasi dalam menentukan tinggi kapal untuk meningkatkan jumlah kehancuran kapal yang semakin banyak. Setelah perang dunia ke-2 pun, perencanaan dalam mengefisiensikan fungsi dari sumber daya pun menjadi hal yang esensial dan terus dikembangkan (Baazara et al., 1977).

Linear programming sebagai salah satu metode optimasi digunakan untuk memaksimalkan atau meminimalkan suatu fungsi tujuan yang berbentuk linear melalui variabel-variabel keputusan terhadap fungsi kendala (i.e., *constraints*) berbentuk linear pula. Metode yang digunakan dalam menyelesaikan *linear programming* dapat dibagi menjadi dua bagian besar yaitu (i) *graph-based method* atau metode berbasis grafik dan (ii) *analytical-based method* yang mencakup umumnya berupa metode *simplex.* Metode *graph* pada *linear programming* memiliki keunggulan dalam merepresentasikan suatu permasalahan. Namun, metode ini dapat digunakan apabila hanya terdapat dua variabel keputusan pada fungsi objektif yang ditetapkan. Metode ini memiliki dua varian prosedur, yakni *isoline profit method* atau *corner point method.*

2.1. Metode *Graph* dengan *Iso Line Profit*

Prinsip dari metode ini adalah membuat *iso line* atau garis-garis yang sama berdasarkan berdasarkan fungsi objektif dan mencari nilai yang paling optimal secara iteratif dari permasalahan yang ada. Langkah-langkah dari metode ini adalah:

1. Memformulasikan permasalahan menjadi fungsi matematika yang mencakup fungsi objektif dan variabel keputusan yang dapat diambil.

2. Mendefinisikan *feasible region* berdasarkan batasan-batasan atau *constraints* dalam fungsi matematika yang digunakan untuk mengambil keputusan.
3. Mengasumsikan *initial solution* (contoh: Z_0), misal $c_1x_1 + c_2x_2 = Z_0$
4. Menggambar *initial contour line* berdasarkan asumsi *initial solution* yang sudah ditentukan.
5. Menggeser *contour line* secara iteratif untuk mendapatkan hasil yang paling optimum dari *feasible region* yang telah didefinisikan.

Contoh Soal 2.1

Seorang investor berencana menyewakan dua gedung parkir yang dimiliki, dengan luas masing-masing gedung adalah 5000 m^2 dan 3000 m^2. Untuk memaksimalkan keuntungannya, investor tersebut menetapkan bahwa penyewa harus menyewa gedung dengan luas minimal 7000 m^2. Harga sewa untuk kedua gedung tersebut adalah Rp. 50.000 per m^2 dan Rp. 60.000 per m^2. Berapa luas dari masing-masing gedung yang akan disewakan untuk memaksimalkan keuntungan investor?

Tahap 1: *Formulasi masalah menjadi model matematika*

Tahap pertama dari penyelesaian masalah optimasi ini adalah dengan memformulasikan permasalahan tersebut dalam bentuk model matematika. Terdapat tiga fitur utama yang harus di identfikasi, yaitu variabel keputusan, fungsi tujuan, dan fungsi kendala.

- Variabel keputusan pada permasalahan tersebut adalah berapa luas yang akan disewakan pada masing-masing gedung. Sehingga anggap bahwa x_1 merepresentasikan luas yang disewakan pada gedung 1, sementara x_2 adalah luas yang akan disewakan pada gedung 2.
- Fungsi tujuan dapat terlihat jelas bahwa investor ingin memaksimalkan keuntungan dari proses menyewakan aset yang dia miliki. Sehingga fungsi tujuannya adalah perkalian dari luas yang disewakan pada gedung 1 dan gedung 2 dikalikan dengan harga sewa pada masing-masing gedung. Hal ini dapat diformulasikan pada Persamaan (2.1).
- Fungsi kendala dari permasalahan ini dapat diidentifikasi pada 3 bagian yaitu
  - Batasan luas total yang dapat disewakan yang harus lebih besar daripada 7000 m^2 , dinyatakan dalam Persamaan (2.2).
  - Batasan luas untuk gedung 1 harus kurang dari sama dengan 5000 m^2 dinyatakan dalam Persamaan (2.3), sementara untuk gedung 2 harus kurang dari sama dengan 3000 m^2 dinyatakan dalam Persamaan (2.4).

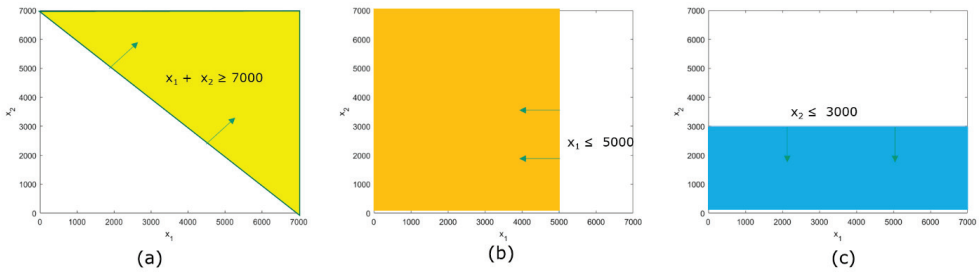
- Batasan bahwa luas gedung yang disewa tidak boleh negatif, atau biasa disebut *non negativity constraint* dinyatakan dalam Persamaan (2.5).

Permasalahan tersebut kemudian jika dituliskan secara lengkap adalah sebagai berikut.

$$Max\ Z = 50\,x_1 + 60\,x_2 \text{ (dalam ribu rupiah)} \quad (2.1)$$
$$x_1 + x_2 \geq 7000 \quad (2.2)$$
$$x_1 \leq 5000 \quad (2.3)$$
$$x_2 \leq 3000 \quad (2.4)$$
$$x_1 \geq 0, x_2 \geq 0 \quad (2.5)$$

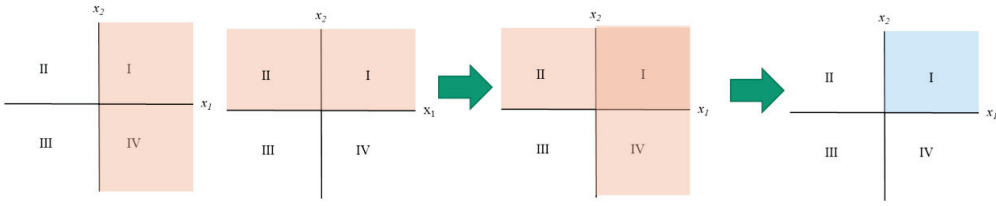
***Tahap 2**: Mendefinisikan area solusi yang mungkin (feasible region)*

Setiap fungsi kendala yang telah ditetapkan sebelumnya akan digunakan untuk mengetahui area solusi yang mungkin (*feasible region*), yang dilihat pada Gambar 2.1 hingga Gambar 2.3.



Gambar 2.1 Fungsi kendala terkait fungsi luas.

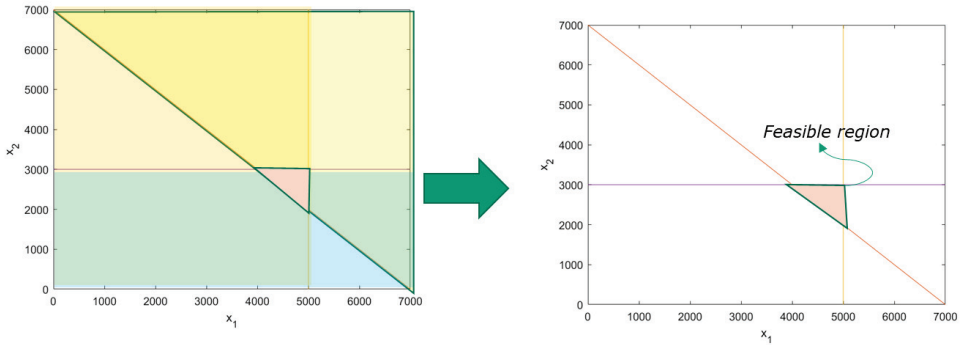
Gambar 2.1 (a) menunjukkan area solusi yang mungkin berdasarkan fungsi kendala total luas yang dapat disewa, dimana harus lebih besar sama dengan 7.000 m^2. Sementara Gambar 2.1 (b) dan (c) mengilustrasikan area solusi yang mungkin untuk fungsi kendala pada gedung 1 dan gedung 2. Selain itu, terdapat pula area solusi yang dibentuk oleh fungsi kendala non negatif (lihat Gambar 2.2). Fungsi kendala non-negatif ini berusaha memastikan bahwa nilai untuk x_1 dan x_2 selalu bernilai positif atau berada pada kuadran 1.



Gambar 2.2 Fungsi kendala non-negatif.

Menggabungkan seluruh fungsi kendala diatas, maka area solusi yang mungkin dari permasalahan ini dapat dilihat pada Gambar 2.3. Area ini sesungguhnya mengambarkan wilayah dari variabel keputusan yang dapat diambil tanpa melanggar batasan ataupun

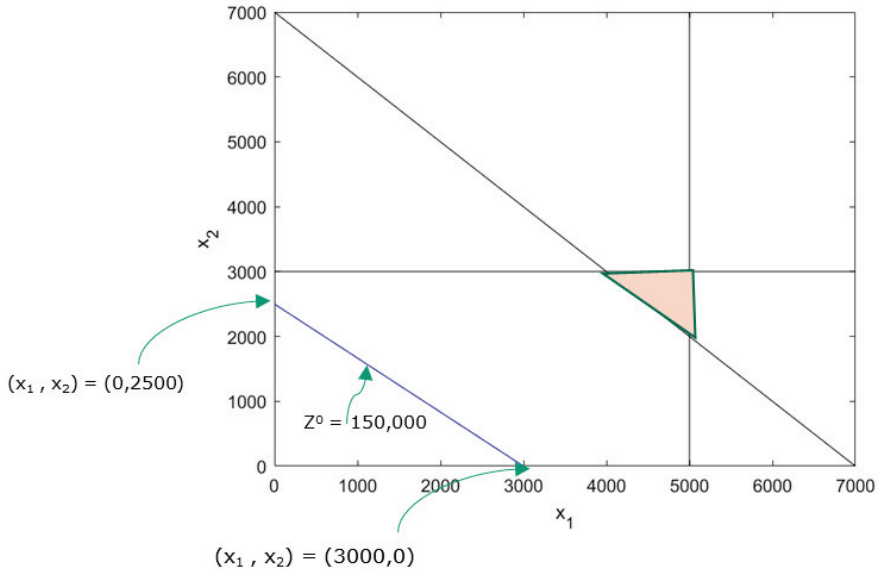
fungsi kendala yang ada. Keputusan di luar area ini dapat dipastikan akan melanggar fungsi kendala, yang artinya solusi yang ditawarkan tidak optimal.



Gambar 2.3 Ilustrasi area solusi yang mungkin (*feasible region*).

***Tahap 3**: Menetapkan solusi awal dan menggambar garis kontur*

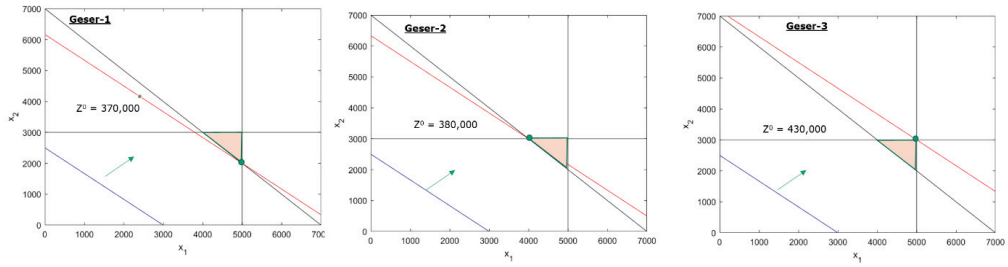
Untuk menetapkan solusi awal, dapat digunakan *initial contour line* yang dibuat dengan mengasumsikan nilai *initial solution* berdasarkan fungsi tujuan. Sebagai contoh, anggap terdapat solusi awal sebesar adalah Rp. 150.000 (dalam ribu), sehingga $50\,x_1 + 60x_2 = 150000$. Jika $x_1 = 0$, maka x_2=3000 sementara jika $x_2 = 0$, maka x_1=2500. Dengan demikian, dapat digambarkan garis kontur dari nilai awal dari fungsi tujuan tersebut seperti yang terlihat pada Gambar 2.4.



Gambar 2.4 Ilustrasi solusi awal dengan garis kontur.

***Tahap 4**: Mengeser garis kontur awal*

Setelah *initial contour line* sudah digambar, maka garis tersebut dapat digeser dengan kemiringan yang sama hingga berpotongan pada titik-titik pada *feasible region,* sehingga didapatkan titik-titik ekstrim.



Gambar 2.5 Ilustrasi proses pergeseran garis kontur awal.

Berdasarkan Gambar 2.5, terlihat bahwa didapatkan tiga titik ekstrim yang akan dievaluasi. Nilai x_1 dan x_2 dari setiap titik ekstrim dapat dimasukkan ke dalam fungsi objektif untuk menghitung nilai *fitness* (Z). Pada proses pergeseran garis yang pertama didapatkan Z untuk titik ekstrim pertama (5000, 2000) sebesar Rp. 370.000 (dalam ribu), dan Z untuk titik ekstrim lainnya (3000, 4000) dan (5000, 3000) sebesar masing-masing Rp. 380.000 and Rp. 430.000 (dalam ribu). Karena permasalahan optimasi ini bermaksud untuk memaksimalkan keuntungan maka solusi yang direkomendasikan adalah pada titik ekstrim kedua yang memberikan nilai *fitness* terbesar yaitu Rp. 430.000 (dalam ribu). Dengan demikian dapat disimpulkan bahwa untuk memaksimalkan keuntungan, investor tersebut harus menyewakan gedung 1 seluas 5.000 m^2 dan gedung 2 sebesar 3.000 m^2.

2.2. *Metode Graph* dengan *Corner Point*

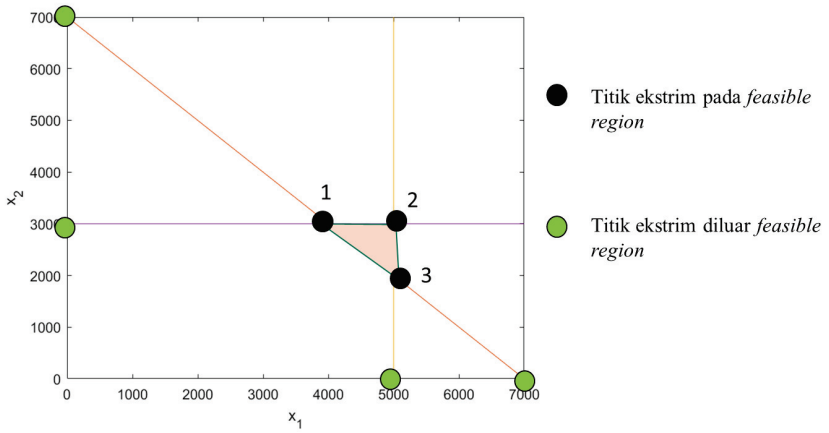
Prinsip dari metode ini adalah mengevaluasi setiap *corner point* untuk mencari nilai yang paling optimal dari permasalahan yang ada. *Corner point* merupakan titik sudut yang terbentuk dari perpotongan garis pada fungsi kendala. Langkah-langkah dari metode ini adalah:

1. Memformulasikan permasalahan menjadi fungsi matematika yang mencakup fungsi objektif dan variabel keputusan yang dapat diambil.
2. Mendefinisikan *feasible region* berdasarkan batasan-batasan dari fungsi kendala *constraints* dalam model matematika yang digunakan untuk mengambil keputusan.
3. Mengidentifikasikan *feasible corner points* pada setiap perpotongan antara *constraints line* pada *feasible region.*

4. Mengevaluasi *fitness value* pada setiap *corner points* berdasarkan fungsi objektif yang sudah ditetapkan.
5. Mencari solusi yang paling optimum berdasarkan hasil *fitness value* yang tertinggi atau terrendah sesuai dengan fungsi objektif yang ditetapkan.

Contoh Soal 2.2

Contoh pada metode ini akan menggunakan permasalahan yang sama dengan **Contoh Soal 2.1** pada bagian sebelumnya, dengan objektif memaksimalkan keuntungan dari luas setiap ruang yang akan disewakan. Langkah yang dilakukan untuk memformulasikan permasalahan menjadi fungsi matematika serta mendefinisikan *feasible region* sama seperti pada contoh pembahasan sebelumnya. Langkah selanjutnya yang perlu dilakukan pada *Corner Point Method* adalah mengidentifikasikan *feasible corner points* yang didapatkan berdasarkan setiap perpotongan antara *constraints line* pada *feasible region. Ekstreme points* yang telah diidentifikasikan dapat dilihat pada Gambar 2.6.



Gambar 2.6 Ilustrasi titik ekstrim.

Titik yang berwarna hijau menandakan perpotongan garis yang berada di luar *feasible region,* dan titik yang berwarna hitam menandakan perpotongan garis yang berada pada *feasible region* atau disebut *feasible extreme points.* Setiap titik pada *feasible extreme point* akan dievaluasi pada fungsi objektif untuk melihat keuntungan maksimum yang dapat dihasilkan. Nilai *fitness* untuk setiap titik tersebut adalah:

- Titik 1 (4000, 3000) = 50 (4000) + 60 (3000)=Rp. 380.000 (dalam ribu)
- Titik 2 (5000, 3000) = Rp. 430.000 (dalam ribu)
- Titik 3 (5000, 2000) = Rp. 370.000 (dalam ribu)

Sesuai fungsi objektif diharapkan untuk memaksimalkan keuntungan, maka solusi yang diambil adalah solusi dengan nilai *fitness Z* yang terbesar, yakni Rp. 430.000 pada titik 2 (5000,3000). Solusi pada metode ini memiliki solusi yang sama seperti solusi *dengan Isoline Profit Method*, yakni untuk memaksimalkan keuntungan pemilik gedung, luas pada gedung 1 yang akan disewakan adalah 5000 m^2 dan luas gedung 2 yang akan disewakan adalah 3000 m^2.

2.3. *Sensitivity Analysis* dengan Metode *Graph*

Setelah memperoleh solusi yang paling optimal, tahapan *sensitivity analysis* dapat dilakukan untuk mengetahui dampak perubahan dari fungsi objektif maupun fungsi kendala terhadap solusi optimal. *Sensitivity analysis* umumnya melihat perubahan solusi optimal terhadap perubahan atas dua hal yaitu (i) perubahan koefisien pada fungsi objektif, (ii) perubahan pada fungsi kendala.

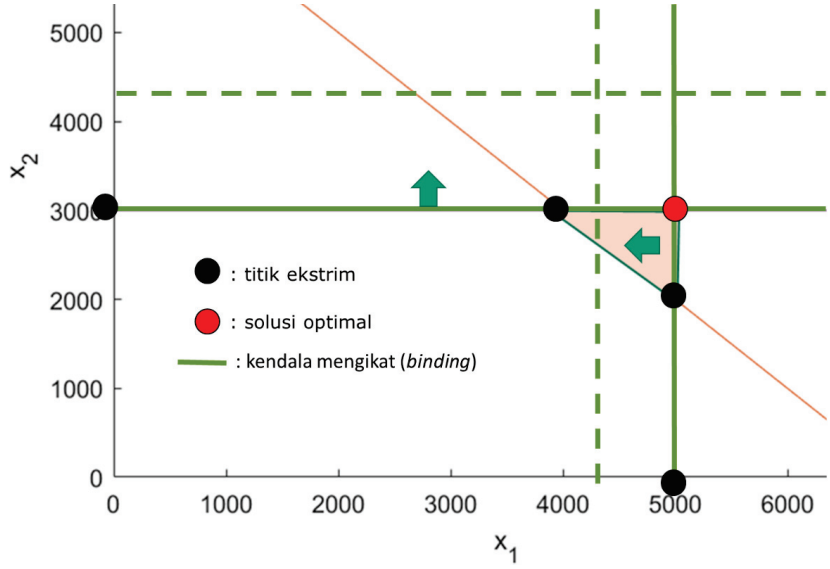
Sensitivity analysis terkait eksplorasi perubahan dari koefisien pada fungsi objektif dilakukan karena jika koefisien pada fungsi objektif berubah, maka akan terjadi perubahan gradien pada *iso line* yang berpotensi menyebabkan perubahan solusi. Sementara pada konteks perubahan pada fungsi kendala *sensitivity analysis* dapat digunakan untuk menginvestigasi seberapa besar dampak dari penambahan atau pengurangan *resources* pada fungsi kendala berpengaruh terhadap perubahan solusi optimal.

Perubahan pada *constraints* dapat dievaluasi pada bagian *binding constraints* dan *non-binding constraints. Binding constraint* adalah *constraint* yang berkaitan langsung dengan solusi optimal, atau sering disebut sebagai fungsi kendala yang menjadi lokasi solusi optimal. Sebaliknya *non-binding constraint* merupakan fungsi kendala yang tidak terkait langsung dengan solusi optimal (lihat Gambar 2.7).

Resources pada fungsi kendala yang terkait terhadap perubahan *binding constraint* diistilahkan sebagai *scarce resource* atau sumber yang langka. *Sensitivity analysis* digunakan untuk mengetahui perubahan terhadap solusi optimal jika terdapat perubahan dari *scarce resource,* baik bersifat penambahan maupun pengurangan. Analisis ini juga dapat digunakan bertujuan untuk mencari penambahan dari *scarce resource* terbaik dalam rangka meningkatkan solusi optimal yang diharapkan.

Pada perubahan *non-binding constraint*, *sensitivity analysis* digunakan untuk mengetahui batasan penambahan atau pengurangan agar tidak mengubah solusi optimal

yang ada. *Resources* pada *non-binding constraint* disebut dengan *abundant resources* atau *redundant resources.* Contoh soal dibawah ini kemudian diberikan dalam rangka memahami lebih jauh penggunaan *sensitivity analysis* pada LP dengan *graph*.



Gambar 2.7 Ilustrasi kendala mengikat (*binding constraint*).

Contoh Soal 2.3

Sebuah perusahaan pemasok aspal beton memproduksi dua jenis tipe aspal beton, yakni aspal beton A dan aspal beton B. Setiap aspal beton memiliki komposisi material, dan keuntungan yang berbeda, seperti yang dapat dilihat pada tabel di bawah ini. Ketersediaan material setiap harinya juga membatasi produksi kedua jenis beton tersebut. Selain itu, juga diketahui bahwa terdapat selisih antara produksi dari aspal beton B dan aspal beton A sebesar 1 ton, dimana produksi aspal beton B selalu lebih besar dibandingkan aspal beton A. Jumlah kebutuhan maksimum aspal beton B per hari adalah 2 ton.

Tabel 2.1 Kebutuhan dan ketersediaan material untuk produksi.

	Kebutuhan Material untuk Produksi 1 ton		**Ketersediaan Material (ton)**
	Aspal Beton A	**Aspal Beton B**	
Material Tipe 1	5	4	20
Material Tipe 2	1	4	8
Profit per Ton (10 juta Rp.)	3	5	-

a. Berapa komposisi produksi aspal beton A dan aspal beton B yang paling optimum untuk memberikan keuntungan maksimal bagi perusahaan?
b. Berapa *range of optimality* yang masih dapat diterima jika terjadi perubahan pada *objective function?*

c. Berapa komposisi produksi aspal beton A dan aspal beton B yang paling optimum jika terjadi penurunan profit pada beton B sebesar 10 juta rupiah/ton (i.e., profit aspal beton B menjadi Rp. 40 juta/ton) ?
d. Berapa komposisi produksi aspal beton A dan aspal beton B yang paling optimum jika ketersediaan material 1 berubah menjadi (turun) 15 ton/hari dan (naik) 30 ton/hari?
e. Berapa komposisi produksi aspal beton A dan aspal beton B yang paling optimum jika ketersediaan material 1 naik 25 ton/hari?
f. Berapa komposisi produksi aspal beton A dan aspal beton B yang paling optimum jika ketersediaan material 2 berubah menjadi (naik) 10 ton/hari dan (turun) 4 ton/hari?
g. Ketersediaan material yang mana yang sebaiknya ditambahkan untuk memberikan keuntungan yang lebih kepada perusahaan?

a. Solusi optimum komposisi aspal beton A dan aspal beton B yang diproduksi

Tahapan pertama dalam menyelesaikan permasalahan tersebut, adalah memformulasikan permasalahan tersebut kedalam model matematika. Tiga fitur utama dari permasalahan optimasi dielaborasi sebagai berikut:

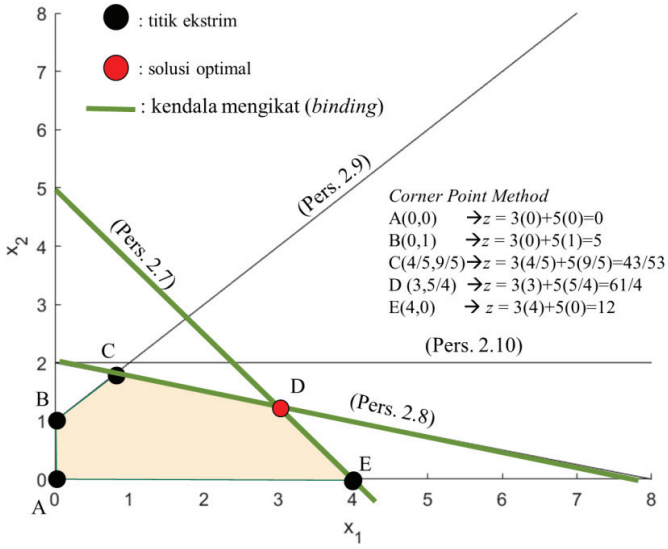
- Variabel keputusan adalah terkait berapa jumlah aspal beton A dan aspal beton B yang diproduksi. Sehingga dapat diasumsikan bahwa x_1 adalah produksi aspal beton A (ton/hari) dan x_2 adalah produksi aspal beton B (ton/hari).
- Fungsi tujuan adalah memaksimalkan keuntungan dari penjualan aspal beton A dan aspal beton B. Sehingga nilai *fitness* dari fungsi tujuan dapat diperoleh dengan mengalikan jumlah produksi aspal A dan B (i.e., x_1, x_2) dan harga jual masing-masing aspal beton. Fungsi tujuan dari permasalahan ini dituliskan pada Persamaan (2.6).
- Fungsi kendala pada permasalahan ini dapat diidentifikasi sebagai berikut:
  - Adanya batasan terkait jumlah material 1 yang digunakan untuk produksi aspal beton A dan aspal beton B. Sehingga batasan tersebut dapat dituliskan pada Persamaan (2.7).
  - Adanya batasan terkait jumlah material 2 yang digunakan untuk produksi aspal beton A dan aspal beton B. Sehingga batasan tersebut dapat dituliskan pada Persamaan (2.8).
  - Selain itu, juga diketahui bahwa terdapat selisih antara produksi dari aspal beton B dan aspal beton A sebesar 1 ton, dimana produksi aspal beton B lebih besar dibandingkan aspal beton A. Sehingga batasan tersebut dapat dituliskan pada Persamaan (2.9).

- Jumlah kebutuhan maksimum aspal beton B per hari adalah 2 ton, dituliskan pada Persamaan (2.10).
- Batasan bahwa jumlah produksi tidak boleh negatif, dituliskan dalam Persamaan (2.11).

Permasalahan di atas akan diformulasikan ke dalam fungsi matematika sebagai berikut.

$$Max\ Z = 3\,x_1 + 5\,x_2 \quad (2.6)$$
$$5x_1 + 4\,x_2 \leq 20 \quad (2.7)$$
$$x_1 + 4x_2 \leq 8 \quad (2.8)$$
$$-x_1 + x_2 \leq 1 \quad (2.9)$$
$$x_2 \leq 2 \quad (2.10)$$
$$x_1 \geq 0, x_2 \geq 0 \quad (2.11)$$

Setiap *constraint* yang ditetapkan di atas dibuat menjadi *feasible region* dengan memplotkan setiap fungsi untuk kemudian diidentifikasi perpotongannya. Area yang dilingkupi oleh perpotongan garis tesebut kemudian didefinisikan sebagai *feasible region* dari permasalahan optimasi, seperti yang dapat dilihat pada Gambar 2.8.



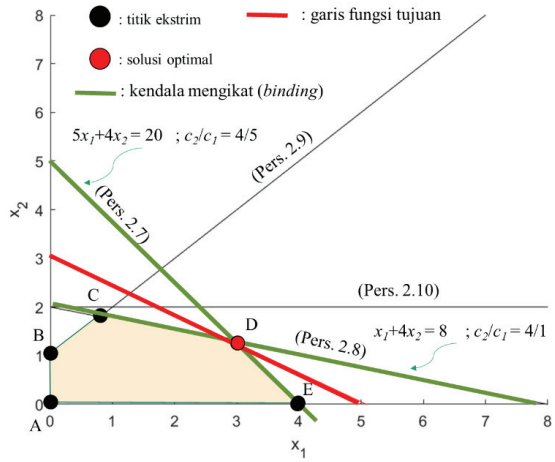
Gambar 2.8 Solusi dari *corner point* method.

Dengan menggunakan metode *corner point method,* maka bisa didapatkan *feasible extreme points* sebanyak 6 titik, yang ditunjukkan pada huruf A, B, C, D, E, dan F (lihat Gambar 2.8). Setiap titik tersebut akan dievaluasi terhadap fungsi tujuannya. Sesuai fungsi tujuan yang diharapkan untuk memaksimalkan keuntungan, maka solusi yang diambil adalah solusi dengan Z yang terbesar, yakni 15,25 (dalam 10 juta rupiah) pada titik D (3, 5/4). Dengan demikian, untuk memaksimalkan keuntungan perusahaan, aspal beton yang

akan diproduksi adalah sebanyak 3 ton/hari untuk aspal beton A dan 5/4 ton/hari untuk aspal beton B.

b. Range of optimality yang masih dapat diterima jika terjadi perubahan pada fungsi tujuan

Perubahan pada *objective function* pada permasalahan ini bisa terjadi ketika adanya perubahan unit keuntungan/harga jual yang dihasilkan dari produksi aspal beton A atau aspal beton B. Sehingga dimungkinkan pada suatu kondisi di mana solusi optimum akan berubah ke titik ekstrim lainnya. *Range* dari perubahan koefisien pada fungsi tujuan yang tidak menyebabkan perubahan titik ekstrim (solusi optimal) disebut *range of optimality*. Range yang masih dapat diterima dapat ditentukan dengan membandingkan koefisien c_2 dibandingkan koefisien c_1 untuk kedua garis *constraint* yang berpotongan pada titik. Ilustrasi perbandingan c_2/c_1 tersebut dapat dilihat pada Gambar 2.9 berikut.



Gambar 2.9 Ilustrasi *range of optimality*.

Dengan demikian, *range of optimality* untuk perubahan pada *objective function* masih diterima apabila terjadi perubahan koefisien pada *objective function* berada pada rentang $\frac{4}{5} \leq \frac{c_2}{c_1} \leq \frac{4}{1}$. Secara visual rentang ini dapat juga dilihat sebagai rentang yang membatasi gradien dari garis yang merepresentasikan fungsi tujuan (i.e., garis merah pada Gambar 2.9)

c. Solusi optimum jika terjadi penurunan harga jual pada produksi aspal beton B

Jika terjadi penurunan profit pada produksi aspal beton B, maka *objective function* menjadi,

$$Max\ Z = 3x_1 + 4x_2 \qquad (2.12)$$

Nilai c_2/c_1 pada *objective function* (nilai *fitness*) tersebut adalah 4/3 dan masih berada pada rentang *range of optimality* pada solusi sebelumnya. Maka dari itu solusi optimum tidak berubah, yakni 3 ton/hari untuk aspal beton A dan 5/4 ton/hari untuk aspal beton B.

d. Solusi optimum jika ketersediaan material 1 berubah menjadi 15 ton/hari dan 30 ton/hari

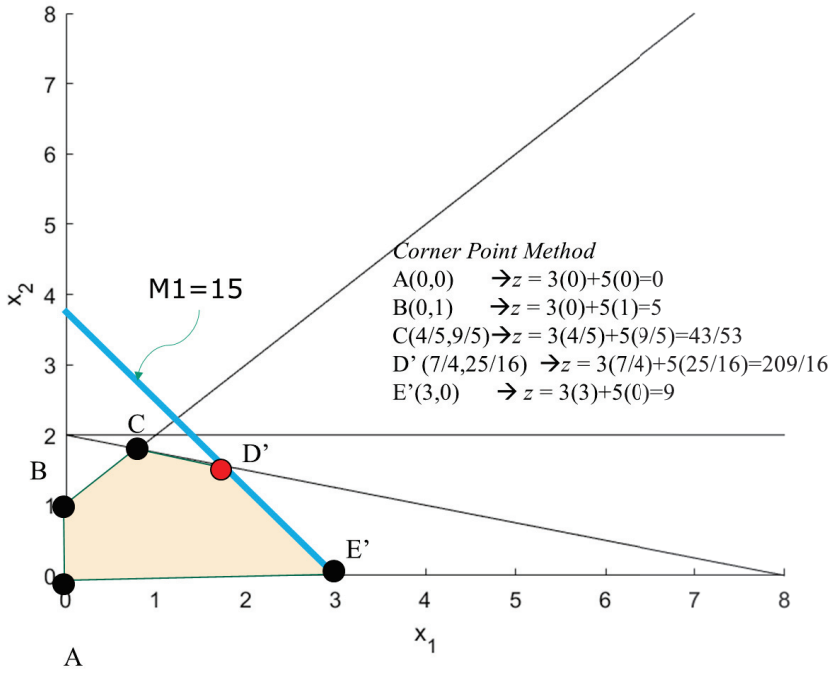
Pada bagian ini akan dievaluasi jika terjadi perubahan dari sisi fungsi kendala, dimana ketersediaan material 1 dapat berubah yang awalnya berjumlah 20 ton/hari. Terdapat dua kemungkinan perubahan, yaitu:

- *ketersediaan material 1 berubah menjadi 15 ton/hari*
- *ketersediaan material 1 berubah menjadi 30 ton/hari*

Pada skenario pertama, dengan berkurangnya material 1 (i.e., M1) menjadi 15 ton/hari, maka *constraint* pada Persamaan (2.4) berubah menjadi,

$$5x_1 + 4\,x_2 \leq 15 \qquad (2.13)$$

Dengan demikian, *feasible region* akan berubah menjadi Gambar 2.10 berikut.



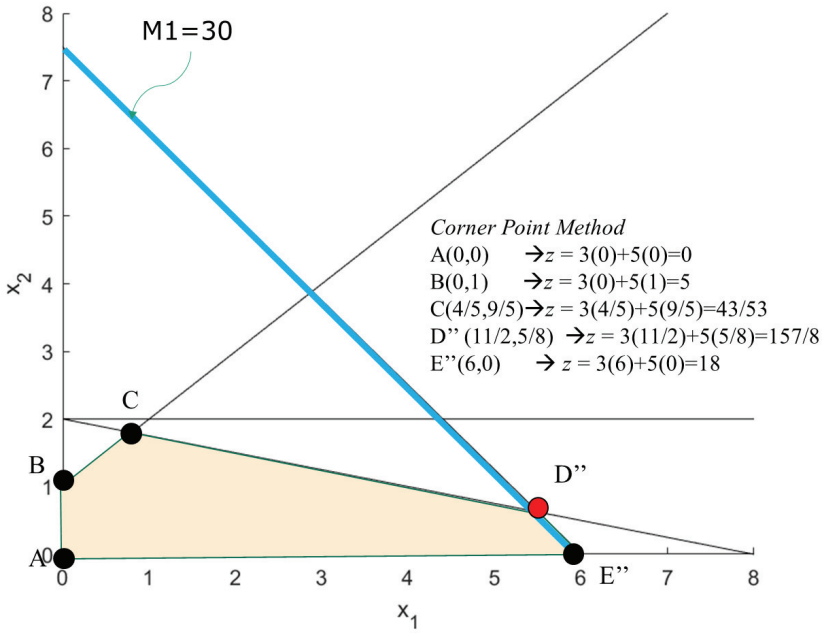
Gambar 2.10 Ilustrasi kasus ketersediaan material 1 berubah menjadi 15.

Dengan menggunakan *corner point method*, titik ekstrim yang berada pada *feasible region* kemudian dievaluasi kembali. Hasil evaluasi pada setiap titik ekstrim diketahui bahwa solusi optimum ikut berubah, yakni $z = \frac{209}{16} \sim 13{,}06$, dengan produksi aspal beton A $x_1 = \frac{7}{4}\ ton/hari$, dan produksi aspal beton B $x_2 = \frac{25}{16}\ ton/hari$.

Sementara pada skenario 2, jika material 1 bertambah menjadi 36, maka *constraint* pada Persamaan (2.4) berubah menjadi,

$$5x_1 + 4\,x_2 \leq 30 \qquad (2.14)$$

Dengan demikian, *feasible region* akan berubah menjadi gambar berikut.



Gambar 2.11 Ilustrasi kasus ketersediaan material 1 berubah menjadi 30.

Berdasarkan kepada *feasible extreme points* yang baru akibat adanya perubahan ketersediaan material pada fungsi kendala, diketahui bahwa solusi optimum ikut berubah, yakni $z = \frac{157}{8} \sim 19{,}63$ .Solusi ini diperoleh dengan melakukan produksi aspal beton A sebesar $x_1 = \frac{157}{8}\ ton/hari$, dan produksi aspal beton B sebesar $x_2 = \frac{5}{8}\ ton/hari$.

e. Solusi optimum jika ketersediaan material 1 bertambah menjadi 25 ton/hari

Pertanyaan kelima pada contoh soal ini berisi tentang perubahan material 1 yang dapat bertambah sebesar 25 ton per hari. Proses yang sama sesungguhnya dapat dilakukan seperti pada contoh bagian d. Meskipun karena sebelumnya telah diketahui perubahan

ketersediaan material menjadi 15 dan 30, maka perubahan ketersediaan di antara rentang tersebut terhadap solusi optimal juga dapat dicari secara langsung menggunakan pendekatan persamaan dua variabel, tanpa harus menggambar *feasibility region* yang baru.

$$aM1 + b = x_1 \tag{2.15}$$

$$solusi\ untuk\ M1 \leq 15, x_1 = \frac{7}{4}, x_2 = \frac{25}{16} \tag{2.16}$$

$$solusi\ untuk\ M1 \leq 30, x_1 = \frac{157}{8}, x_2 = \frac{5}{8}, \tag{2.17}$$

$$a\ (15) + b = \frac{7}{4}\ ;\ a(30) + b = \frac{157}{8} \tag{2.18}$$

$$\therefore a = \frac{143}{120}, b = -\frac{129}{8} \tag{2.19}$$

$$x_1 = \frac{143}{120}M1 - \frac{129}{8} \tag{2.20}$$

$$aM1 + b = x_2 \tag{2.21}$$

$$a\ (15) + b = \frac{25}{16}; a(30) + b = \frac{5}{8} \tag{2.22}$$

$$\therefore a = -\frac{1}{16}, b = \frac{10}{4} \tag{2.23}$$

$$x_2 = -\frac{1}{16}M1 + \frac{10}{4} \tag{2.24}$$

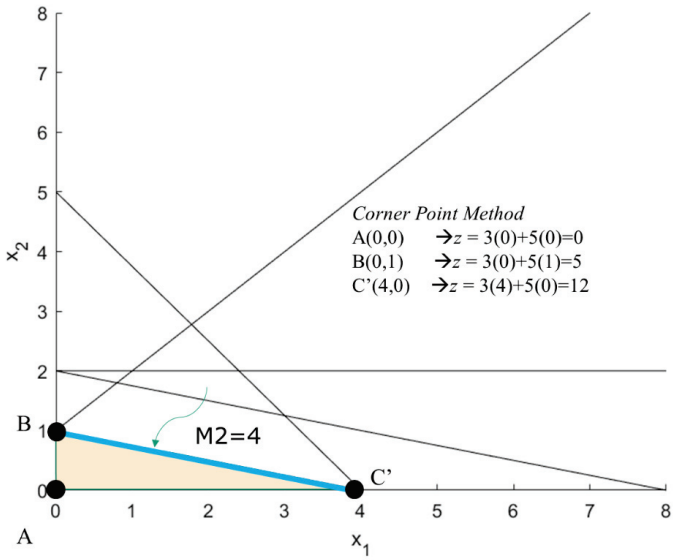
Dengan menggunakan Persamaan (2.15) hingga (2.24) dapat dihitung perubahan batasan ketersediaan ketika berubah menjadi 25 (M1 = 25), maka produksi beton A $x_1 = \frac{53}{4}\ ton/hari$, dan produksi beton B $x_2 = \frac{15}{16}\ ton/hari$.

f. Solusi optimum jika ketersediaan material 2 bertambah menjadi 10 ton/hari dan menurun menjadi 4 ton/hari.

Pada contoh ini, perubahan terjadi pada material 2, dimana ketersediaan material 2 yang awalnya adalah 6 ton/hari berkurang menjadi 4 ton/hari, maka *constraint* Persamaan (2.5) berubah menjadi,

$$x_1 + 4x_2 \leq 4 \tag{2.25}$$

Dengan demikian, *feasible region* akan berubah menjadi gambar berikut.



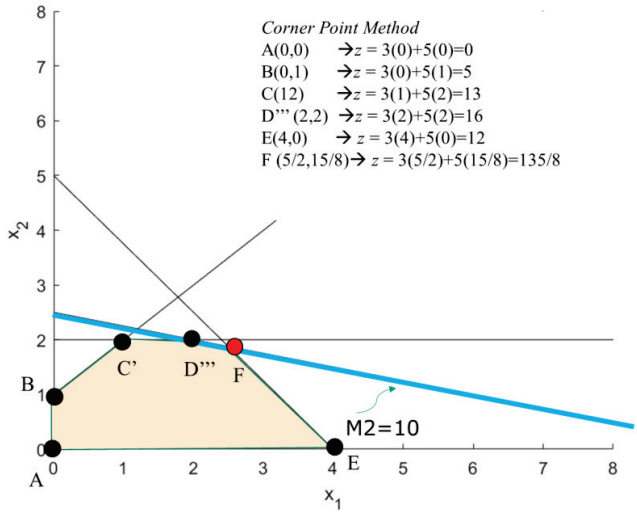
Gambar 2.12 Ilustrasi kasus ketersediaan material 2 berubah menjadi 4.

Dengan menggunakan pendekatan *corner point*, diperoleh bahwa solusi optimum ikut berubah, yakni dengan produksi aspal beton A $x_1 = 4\ ton/hari$, dan tanpa produksi aspal beton B $x_2 = 0\ ton/hari$ dan i $z = 12$.

Pada kasus lain, dianggap bahwa ketersediaan material 2 dapat bertambah menjadi 6 2/3, maka *constraint* pada Persamaan (2.5) berubah menjadi,

$$x_1 + 4\,x_2 \leq 10 \qquad (2.26)$$

Sehingga *feasible region* akan berubah menjadi Gambar 2.13.



Gambar 2.13 Ilustrasi kasus ketersediaan material 2 berubah menjadi 10.

Perubahan dari area solusi ini kemudian menyebabkan perubahan pada solusi optimum, yakni produksi aspal beton A $x_1 = \frac{5}{2}\frac{ton}{hari}$, dan produksi aspal beton B $x_2 = \frac{15}{8}\frac{ton}{hari}$ dengan nilai fitness $z = \frac{135}{8} \sim 16{,}87$.

g. Material yang sebaiknya ditambahkan untuk memberikan keuntungan lebih.

Untuk menjawab pertanyaan terkait material mana yang sebaiknya ditambah dalam rangka meningkatkan keuntungan dapat dilakukan analisis *unit worth of resource* yang dihitung dengan persamaan:

$$y_i = \frac{Perubahan\ nilai\ fitness}{Perubahan\ resources - i\ pada\ constraint} \tag{2.27}$$

Persamaan ini memberikan informasi perubahan ketersediaan mana pada fungsi kendala yang akan memberikan perubahan terbesar kepada nilai *fitness*. Semakin besar perubahan yang terjadi pada nilai *fitness* akibat adanya perubahan ketersediaan sebuah variabel-*i* (material dalam kasus ini), maka semakin *worth* material-*i* itu untuk tetap dijaga atau ditingkatkan ketersediaannya. Meskipun penggunaan *worth analysis* ini umumnya hanya berguna pada rentang yang kecil. Jika rentang analisis cukup besar nilai *worth* terkadang tidak dapat diterima secara implementatif.

Perubahan dari *optimum fitness value* dan *addition of resources* didapatkan dari hasil yang didapat pada jawaban pada poin d dan f sebelumnya. Solusi optimum pada permasalahan awal ditunjukkan pada titik D pada Gambar 2.8. Ketika terjadi perubahan pada material 1, maka perubahan terjadi dari titik D menjadi D'(Gambar 2.10) atau D'' (Gambar 2.11), dan ketika terjadi perubahan pada material 2, maka perubahan terjadi dari titik D menjadi C' (Gambar 2.12) atau F (Gambar 2.13). Selanjutnya perhitungan *unit worth of resource* dapat dihitung sebagai berikut.

$$y_1 = \frac{z_{D''} - z_D}{M1_{D''} - M1_D} \tag{2.28}$$

$$y_1 = \frac{\frac{157}{8} - \frac{61}{4}}{30 - 20} \tag{2.29}$$

$$y_1 = \frac{33}{80} \sim 0{,}41\ (dalam\ 10\ juta\frac{rupiah}{ton} M1) \tag{2.30}$$

$$y_2 = \frac{z_F - z_D}{M2_F - M2_D} \tag{2.31}$$

$$y_2 = \frac{\frac{135}{8} - \frac{61}{4}}{10 - 8} \tag{2.32}$$

$$y_2 = \frac{13}{16} \sim 0{,}81\ (dalam\ 10\ juta \frac{rupiah}{ton} M2) \tag{2.33}$$

Berdasarkan perhitungan di atas, dapat terlihat bahwa *unit worth of resource* dari material 2 memiliki nilai yang lebih tinggi dibandingkan material 1. Perubahan per unit material 1 akan menyebabkan terjadinya perubahan sebesar 0,41 pada nilai *fitness*, sementara pada material 2 0,81. Sebagai contoh, penurunan (atau kenaikan) ketersediaan 1 ton material 1 akan menyebabkan pengurangan (atau kenaikan) nilai *fitness* (keuntungan) sebesar (0,41 x 10 juta). Sementara penurunan (atau kenaikan) ketersediaan 1 ton material 2 akan menyebabkan pengurangan (atau kenaikan) nilai *fitness* (keuntungan) sebesar (0,81 x 10 juta). Sehingga material 2 dapat disimpulkan lebih 'bernilai' dibandingkan material 1 dari sisi perubahan terhadap nilai *fitness*. Oleh karenanya, sangat penting untuk menjaga ketersediaan dari material 2, ataupun meningkatkan ketersediaan dari material 2.

Pembaca yang tertarik untuk melihat penjelasan dalam bentuk audio-visual terkait Bab ini dapat mengunjungi playlist youtube berikut:

https://bit.ly/Playlist-LPGraph

3. *Linear programming* dengan Metode *Simplex*

3.1. Pengantar Metode *Simplex*

Metode *simplex* merupakan salah satu metode dalam menyelesaikan *linear programming*. Metode ini mampu menganalisis lebih dari dua variabel keputusan yang tidak dapat dilakukan pada metode *graph*. Konsep yang dilakukan pada metode *simplex* ini adalah dengan mengevaluasi setiap *corner point* secara analitik untuk mendapatkan solusi optimal yang diharapkan. Terdapat beberapa cara dalam menuliskan metode *simplex*, meskipun pada buku ini menggunakan pendekatan minimasi dengan dua syarat berikut, yakni:

1. Semua fungsi kendala pada sisi kanan harus bernilai positif.
2. Semua variabel harus bernilai positif.

Bentuk dari metode *simplex* dapat dilihat pada persamaan di bawah.

$$\min z = c'x \tag{3.1}$$

$$Ax = b \tag{3.2}$$

$$x \geq 0 \tag{3.3}$$

$$b \geq 0 \tag{3.4}$$

Dimana A, x, dan b merupakan vektor matriks yang didefinisikan sebagai berikut.

$$A = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ a_{21} & \cdots & a_{2n} \\ a_{m1} & \cdots & a_{mn} \end{bmatrix}, x = \begin{bmatrix} x_1 \\ \cdots \\ x_n \end{bmatrix}, b = \begin{bmatrix} b_1 \\ \cdots \\ b_m \end{bmatrix}, n > m \tag{3.5}$$

m : jumlah fungsi kendala

n : jumlah variabel keputusan

x : vektor untuk variabel keputusan

A : matriks m x n

b : vektor untuk fungsi kendala

Sesuai dengan bentuk standar pada Persamaan (3.1) hingga (3.4), *simplex* mensyaratkan penggunaan bentuk persamaan dalam fungsi kendala. Sehingga bentuk fungsi kendala yang berupa pertidaksamaan (i.e., kurang dari/lebih dari) harus di ubah menjadi bentuk persamaan. Bagian sebelah kanan dari bentuk persamaan (i.e., b) dapat

didefinisikan sebagai batas atau ketersediaan sumber daya yang ada. Bagian sebelah kiri dari bentuk persamaan dapat didefinisikan sebagai penggunaan sumber daya tersebut. Terdapat dua variabel yang digunakan dalam mengubah bentuk pertidaksamaan menjadi bentuk persamaan yaitu,

- Jika terdapat pertidaksamaan berbentuk kurang dari sama dengan (≤), maka selisih antar nilai pada bagian sebelah kanan dan sebelah kiri disebut dengan *slack amount of resource* atau sumber daya yang tidak digunakan. *Slack variable* digunakan untuk mentransformasikan pertidaksamaan berbentuk (≤) menjadi bentuk persamaan.

$$Ax \leq b \tag{3.6}$$

Persamaan (3.6) ditransformasikan menjadi Persamaan (3.7) dan Persamaan (3.8)

$$Ax + x_p = b \tag{3.7}$$

$$x_p \geq 0 \tag{3.8}$$

dimana,

x_p : *slack variable*

- Jika terdapat pertidaksamaan berbentuk lebih besar sama dengan (≥), maka selisih antar nilai pada sebelah kiri dan sebelah kanan disebut dengan *surplus* atau sumber daya yang berlebih. *Surplus variable* kemudian digunakan untuk mentransformasikan pertidaksamaan berbentuk lebih besar sama dengan (≥) menjadi bentuk persamaan.

$$Ax \geq b \tag{3.9}$$

Sehingga Persamaan (3.9) ditransformasikan menjadi Persamaan (3.10) & Persamaan (3.11)

$$Ax - x_r = b \tag{3.10}$$

$$x_r \geq 0 \tag{3.11}$$

dimana,

x_r = *surplus variable*

3.2. Konsep Dasar Perhitungan *Simplex*

Solusi yang terdapat pada *simplex* dapat terbagi menjadi solusi basis dan non basis. Solusi basis merepresentasikan *corner point* yang akan dievaluasi, dimana jumlahnya sama

dengan jumlah fungsi kendala. Sementara solusi non basis adalah kandidat solusi selain solusi basis yang tidak akan dievaluasi nilai fungsi objektifnya. Sehingga pada metode *simplex* variabel yang merupakan solusi non basis selalu dianggap memiliki nilai sama dengan nol. Jumlah dari variabel non basis dapat ditentukan dari selisih jumlah variabel (n) dikurangi dengan jumlah variabel basis (m) atau ditulis dengan $(n-m)$. Ilustrasi solusi basis dan non basis dapat dituliskan pada fungsi sebagai berikut.

$$Ax = [A_B \quad A_N] \begin{bmatrix} x_B \\ x_N \end{bmatrix} = b \tag{3.12}$$

$$A_B x_B + A_N x_N = b \tag{3.13}$$

x_B : vektor basis

x_N : vektor non basis

Dengan adanya variabel basis dan non basis, maka fungsi objektif dapat ditulis dengan memasukkan variabel basis dan non basis, yakni:

$$z = c'x \tag{3.14}$$

$$z = c'_B\, x_B + c'_N\, x_N \tag{3.15}$$

Berdasarkan persamaan pada *constraint* dan fungsi objektif di atas, maka dapat dibentuk menjadi sistem persamaan linear dalam matriks.

$$\begin{bmatrix} A_B & A_N \\ c_B^T & c_N^T \end{bmatrix} \begin{bmatrix} x_B \\ x_N \end{bmatrix} = \begin{bmatrix} b \\ z \end{bmatrix} \tag{3.16}$$

$$\begin{vmatrix} A_B & A_N & b \\ c_B^T & c_N^T & z \end{vmatrix} \tag{3.17}$$

Untuk menyelesaikan permasalahan tersebut, dapat dilakukan dengan menggunakan teknik-teknik untuk menyelesaikan persamaan linear dengan matrik. Salah satu tekniknya adalah *row operation*. Untuk mendapatkan solusi, *row operation* bekerja sedemikian sehingga matrik tersebut dapat membentuk seperti matrik yang memenuhi syarat-syarat di dalam kaidah *reduced row echelon form.* Adapun tahapannya adalah sebagai berikut:

Step 1: Membagi ruas pertama dengan A_B agar kolom 1 baris 1 bernilai 1 atau I (dalam konteks matriks).

$$\begin{vmatrix} I & A_B^{-1}A_N & A_B^{-1}b \\ c_B^T & c_N^T & z \end{vmatrix} \tag{3.18}$$

Step 2: Menambahkan baris 2 dengan baris 1 yang dikalikan $-c_B^T$ agar kolom 1 baris 2 bernilai 0.

$$\begin{vmatrix} I & A_B^{-1}A_N & A_B^{-1}b \\ c_B^T - c_B^T & c_N^T - c_B^T A_B^{-1}A_N & z - c_B^T A_B^{-1}b \end{vmatrix} \tag{3.19}$$

Selanjutnya perlu didefinisikan nilai c_l^T dan z^k dengan,

$$c_l^T = c_N^T - c_B^T A_B^{-1}A_N \tag{3.20}$$

$$z^k = c_B^T A_B^{-1}b \tag{3.21}$$

Dengan demikian, maka hasil matriks dapat ditulis sebagai berikut.

$$\begin{vmatrix} I & A_B^{-1}A_N & A_B^{-1}b \\ c_B^T - c_B^T & c_l^T & z - z^k \end{vmatrix} \tag{3.22}$$

Maka dari itu, bentuk solusi persamaan linear dapat ditulis sebagai berikut.

$$\begin{bmatrix} I & A_B^{-1}A_N \\ 0^T & c_l^T \end{bmatrix} \begin{bmatrix} x_B \\ x_N \end{bmatrix} = \begin{bmatrix} A_B^{-1}b \\ z - z^k \end{bmatrix} \tag{3.23}$$

Berdasarkan matriks di atas, maka dapat dihasilkan Persamaan (3.26) berikut melalui perkalian matriks.

$$x_B + A_B^{-1}A_N x_N = A_B^{-1}b \tag{3.24}$$

$$x_B = A_B^{-1}b - A_B^{-1}A_N x_N \tag{3.25}$$

$$x_B = A_B^{-1}(b - A_N x_N) \tag{3.26}$$

Untuk mengevaluasi *corner point,* maka seluruh variabel non basis harus dibuat menjadi nilai 0, maka $x_N = 0$. Persamaan lain yang dihasilkan dari matriks untuk baris kedua adalah sebagai berikut.

$$0^T x_B + c_l^T x_N = z - z^k \tag{3.27}$$

$$c_l^T x_N = z - c_B^T A_B^{-1}b \tag{3.28}$$

$$z = c_l^T x_N + c_B^T x_B \tag{3.29}$$

Persamaan (3.29) menunjukkan fungsi objektif, di mana untuk memaksimalkan nilai *z*, maka nilai $c_l^T x_N$ harus dibuat tidak negatif agar tidak mengurangi nilai *z*. Dengan demikian, perlu dicari suatu konfigurasi di mana nilai c_l^T tidak negatif, sehingga bisa didapatkan hasil yang maksimal. Kondisi ini akhirnya menjadi salah satu kriteria optimalitas dalam metode *simplex* . Jika $c_l^T x_N$ masih mengandung nilai negatif, maka solusi

yang dihasilkan masih dianggap belum optimal. Selain itu, dengan menggunakan persamaan sebelumnya, dapat diturunkan *augmented matrix* sebagai berikut (lihat Persamaan (3.30) – (3.32)).

$$\begin{vmatrix} I & A_B^{-1}A_N & A_B^{-1}b \\ 0^T & c_l^T & z - z^k \end{vmatrix} \tag{3.30}$$

$$\begin{bmatrix} I & A_B^{-1}A_N \\ 0^T & c_l^T \end{bmatrix} \begin{bmatrix} x_B \\ x_N \end{bmatrix} = \begin{bmatrix} A_B^{-1}b \\ z - z^k \end{bmatrix} \ (dikali\ A_B) \tag{3.31}$$

$$\begin{bmatrix} A_B & A_N \\ 0^T & c_l^T \end{bmatrix} \begin{bmatrix} x_B \\ x_N \end{bmatrix} = \begin{bmatrix} b \\ z - z^k \end{bmatrix}$$

$$\begin{aligned} &dimana \\ &(i)\ c_l^T = c_N^T - c_B^T A_B^{-1} A_N; \\ &(ii)\ z^k = c_B^T A_B^{-1} b = c_B^T x_B \end{aligned} \tag{3.32}$$

Table *augmented matrix* akan tetap berada dalam kondisi optimal jika kedua kondisi ini terpenuhi (Persamaan (3.33) & (3.34)),

$$(i)\ c_l^T \geqslant 0 \tag{3.33}$$

$$c_l^T = c_N^T - c_B^T A_B^{-1} A_N$$

$$c_N^T - c_B^T A_B^{-1} A_N \geqslant 0$$

$$(ii)\ A_B^{-1} b \geqslant 0 \tag{3.34}$$

3.3. *Simplex* Satu Fase

Simplex satu fase merupakan bentuk metode *simplex* yang hanya dapat diaplikasikan untuk permasalahan optimasi sederhana. Permasalahan sederhana tersebut artinya hanya memiliki 1 jenis fungsi kendala, yaitu berbentuk pertidaksamaan kurang dari sama dengan (≤). Jika permasalahan optimasi memiliki fungsi kendala yang berbentuk pertidaksamaan lebih dari sama dengan, atau bentuk persamaan, maka diperlukan *simplex* dua fase. Pada permasalahan optimasi dengan *simplex* satu fase, terdapat 3 syarat kondisi optimal yang akan memandu metode *simplex* dalam menemukan solusi optimal, yaitu:

1. Jika seluruh c_l^k bernilai positif, maka solusi basis merupakan solusi yang optimal.
2. Jika masih terdapat c_l^k yang bernilai negatif, maka solusi basis bukan merupakan optimal, maka perlu dilakukan proses iterasi kembali dengan mengganti solusi non basis sebelumnya menjadi solusi basis berdasarkan nilai terkecil dari c_l^k.

3. Penentuan variabel basis yang harus dikeluarkan dapat dilakukan dengan *minimum ratio test* (*MRT*). Jika koefisien dari variabel basis adalalah 0 atau negatif, maka dapat diterapkan *no limit ratio.*

Berdasarkan ketiga syarat tersebut, maka prosedur umum yang dapat digunakan pada metode *simplex* satu fase adalah sebagai berikut.

1. Menentukan solusi inisial x_0
2. Mengecek solusi inisial x_0 dan dievaluasi terhadap syarat kondisi optimal.
3. Jika kondisi optimal sudah ditemukan, maka solusi inisial digunakan sebagai solusi optimal. Jika belum ditemukan, maka perlu ditentukan solusi baru x_k, lalu tahap 2 dan 3 diulang secara iteratif hingga mendapatkan kondisi optimal.

Contoh Soal 3.1

Suatu maskapai penerbangan perintis mendapatkan izin untuk melayani sebuah rute perjalanan. Rute dengan tarif Rp. 1,000,000 per penumpang ini dapat dilalui dengan menggunakan dua jenis pesawat yang berbeda, yaitu dengan kapasitas 19 penumpang dan 14 penumpang. Untuk melayani rute penerbangan ini, pesawat berkapasitas lebih besar membutuhkan 15 KL avtur, sementara pesawat lainnya hanya membutuhkan 12 KL avtur dengan jumlah avtur yang tersedia untuk rute ini adalah 30 KL per minggu. Selain itu, pesawat dengan kapasitas yang lebih besar merupakan pesawat yang lebih baru memiliki biaya pemeliharaan sebesar 5 juta rupiah per perjalanan, sementara pesawat lainnya 7 juta rupiah per perjalanan. Sementara perusahaan hanya menganggarkan maksimal 25 juta rupiah per minggu untuk biaya perawatan. Perusahaan tersebut bermaksud memaksimalkan keuntungan. Sehingga berapakah jumlah perjalanan per minggu yang harus diputuskan oleh maskapai tersebut untuk masing-masing jenis pesawat. Asumsikan bahwa pesawat akan selalu penuh.

Tahap pertama dalam menyelesaikan permasalahan optimasi adalah memformulasikan permasalahan tersebut ke dalam model matematika dengan mengidentifikasikan fitur utama optimasi.

- Variabel keputusan, adalah jumlah perjalanan perminggu untuk pesawat kapasitas 19 penumpang (x_1) dan pesawat kapasitas 14 penumpang (x_2)
- Fungsi tujuan, adalah memaksimalkan keuntungan yang dapat diformulasikan pada persamaan berikut.

$$Max\ Z = 19\ x_1 + 14x_2 \text{ (dalam juta rupiah)} \quad (3.35)$$

- Fungsi kendala, terdapat beberapa fungsi kendala, yaitu
  - Batasan biaya perawatan yang dianggarkan yang direpresentasikan pada Persamaan (3.36)
  - Batasan *supply* avtur per minggu, yang direpresentasikan pada Persamaan (3.37).
  - Batasan non negatif, yang direpresentasikan pada Persamaan (3.38).

Agar dapat diselesaikan dengan *simplex* satu fase, maka perlu dipastikan terlebih dahulu bahwa bentuk *objective function* adalah meminimalkan masalah. Terlihat bahwa permasalahan ini masih berupa permasalahan maksimasi, sehingga perlu diubah terlebih dahulu menjadi permasalahan minimalisasi. Bentuk *objective function* ini dapat dilihat pada Persamaan (3.35), yang kemudian ditransformasikan dalam bentuk negatif pada Persamaan (3.36). Hal lain harus dipastikan adalah tanda di sebelah kanan sudah positif, dan terdapat *non negativity constraint.* Hal ini sudah dilingkupi oleh formulasi diatas.

Selanjutnya *constraint* akan ditransformasi menjadi bentuk persamaan dengan menambahkan *slack variable* x_3 untuk *constraint* pertama dan x_4 untuk *constraint* kedua. Dengan demikian, didapatkan persamaan berikut.

$$Min\ Z = -19\ x_1 - 14x_2 \quad (3.36)$$

$$15x_1 + 12x_2 + x_3 = 30 \quad (3.37)$$

$$5x_1 + 7x_2 + x_4 = 25 \quad (3.38)$$

$$x_1 \geq 0, x_2 \geq 0, x_3 \geq 0, x_4 \geq 0 \quad (3.39)$$

Pada prinsipnya *initial basic solution* dapat ditentukan secara bebas, meskipun pada umumnya dimulai dengan menetapkan *slack variable* x_3 dan x_4 . Proses ini dianggap memudahkan karena nilai koefisien sudah sama dengan satu. *Initial basic solution* tersebut dapat ditulis pada persamaan di bawah. Dengan mengasumsikan x_3 dan x_4 sebagai solusi basis, maka x_1 dan x_2 adalah solusi non basis yang bernilai 0, maka pada solusi awal tersebut dapat diketahui bahwa, $x_3 = 30, x_4 = 25$.

$$x_B^0 = \begin{bmatrix} x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 30 \\ 25 \end{bmatrix} \quad (3.40)$$

Augmented matrix dari *initial basic* tersebut dapat dibuat dalam tabel berikut sesuai dengan koefisien pada setiap persamaan pada *objective function* dan *constraint* sebelumnya.

Tabel 3.1 Matriks untuk *simplex* iterasi-1.

x_B	x_1	x_2	x_3	x_4	b
x_3	15	12	1	0	30
x_4	5	7	0	1	25
c_l	-19	-14	0	0	(Z-0)

Berdasarkan matriks di atas, masih terdapat nilai c_l bernilai negatif, sehingga solusi belum optimal. Dengan demikian, perlu ditentukan solusi basis yang baru. Solusi basis yang baru menjadi lebih mudah untuk ditentukan dengan menggunakan beberapa istilah berikut:

- *pivot column* sebagai kolom yang merepresentasikan variabel yang akan dijadikan solusi basis,
- *pivot row* sebagai baris yang merepresentasikan variabel yang akan dikeluarkan dari solusi basis, dan
- *pivot number* yang merupakan perpotongan antara *pivot column* dan *pivot row.*

Ketiga istilah tersebut kemudian digunakan untuk memudahkan proses pencarian (i) solusi basis mana yang akan keluar dan (ii) solusi non basis mana yang akan masuk. Pada kasus sebelumnya dapat ditentukan nilai *pivot column* dengan memilih nilai c_l dengan nilai yang paling kecil. Dalam hal ini adalah -19 pada kolom x_1, dengan demikian solusi non basis x_1 akan masuk pada solusi basis. Sementara solusi basis saat ini x_3 atau x_4 salah satunya akan dikeluarkan dari solusi basis. Pemilihan solusi basis yang akan dikeluarkan dapat dilakukan dengan menggunakan *pivot row*. *Pivot row* ini ditentukan dengan metode *minimum ratio test* (MRT) dengan melihat ratio b/a terkecil, dimana a adalah nilai koefisien pada *pivot column*. Nilai b/a yang terkecil adalah sebesar 2 yang berada pada baris x_3. Sehingga baris yang akan dikeluarkan dari solusi basis adalah x_3 yang kemudian akan digantikan dengan solusi non basis x_1 (lihat Tabel 3.2).

Tabel 3.2 Matriks dengan *pivot column*, *pivot row*, dan *pivot number*.

x_B	x_1	x_2	x_3	x_4	b	b/a
x_3	15	12	1	0	30	2
x_4	5	7	0	1	25	5
c_l	-19	-14	0	0	(Z-0)	

Keterangan: *highlight* merah: *pivot row*, *kuning: pivot column*, hijau: *pivot number*.

Berdasarkan tabel tersebut, juga didapatkan *pivot number,* yang merupakan perpotongan antara *pivot column* dan *pivot row,* pada kasus ini adalah 15. Selanjutnya, nilai

pada *pivot number* harus dibuat menjadi 1 dengan membagi *pivot row* dengan *pivot number* seperti yang ditunjukkan pada Tabel 3.3. Untuk memudahkan proses perhitungan, diberikan pula label R1 untuk baris satu yang sudah dibagi dengan *pivot number*. Selain itu, x_1 pun juga akan masuk ke dalam solusi basis menggantikan x_3.

Tabel 3.3 Matriks setelah dibagi dengan *pivot number.*

x_B	x_1	x_2	x_3	x_4	b	b/a	
x_1	1	0,8	0,067	0	2	2	*R1*
x_4	5	7	0	1	25	5	*R2*
c_l	-19	-14	0	0	(Z-0)		*R3*

Keterangan: *highlight* merah: *pivot row*, *kuning*: *pivot column*, hijau: *pivot number*.

Setelah itu, nilai pada *pivot column* selain *pivot number* harus dibuat menjadi 0 dengan mengoperasikan setiap baris terhadap *pivot row* yang baru. Pengoperasian tersebut dapat ditulis sebagai berikut.

$$R2' = R2 - 5 \times R1 \tag{3.41}$$

$$R3' = R3 + 19 \times R1 \tag{3.42}$$

Dengan pengoperasian tersebut, maka bisa didapatkan *augmented matrix* yang baru, yang dapat dilihat pada tabel berikut.

Tabel 3.4 Matriks dengan evaluasi c_l.

x_B	x_1	x_2	x_3	x_4	b	
x_1	1	0.8	0.07	0	2	*R1'*
x_4	0	3	-0.33	1	15	*R2'*
c_l	0	1.2	1.27	0	(Z+38)	*R3'*

Pada hasil *augmented matrix* di atas, nilai c_l seluruhnya sudah bernilai positif, sehingga dapat disimpulkan bahwa solusi sudah optimum. Dengan demikian, solusi basis pada permasalahan di atas adalah x_1 dengan nilai 2 dan x_4 dengan nilai 15. Oleh karena itu, solusi dari permasalahan di atas adalah $x_1 = 2$ dan $x_4 = 15$, dengan $Z = -38$.

Atas dasar tersebut, dapat disimpulkan bahwa penerbangan untuk pesawat tipe 19 penumpang adalah 2 kali seminggu. Sementara potensi pendapatan jika penumpang penuh adalah 38 juta per minggu.

3.4 *Simplex* Dua Fase

Berbeda dengan *simplex* satu fase, metode *simplex* dua fase dapat diaplikasikan untuk kondisi fungsi kendala yang memiliki bentuk persamaan (=) maupun pertidaksamaan lebih besar sama dengan atau kurang dari sama dengan ($\geq$ atau $\leq$). Seperti yang telah

dijelaskan sebelumnya, pada kondisi batasan lebih besar sama dengan, maka perlu ditambahkan *surplus variable.* Namun, penambahan *surplus variable* yang dilakukan secara langsung pada variabel basis dapat menyebabkan kondisi *non negativity constraint* dari variabel basis tidak dapat dipenuhi. Dengan demikian, perlu ditambahkan suatu *artificial variable* pada pertidaksamaan lebih besar sama dengan untuk mengatasi kondisi ini.

Artificial variables ditambahkan pada fungsi pertidaksamaan lebih besar sama dengan atau fungsi kendala dengan bentuk persamaan. *Artificial variables* merupakan variabel baru yang tidak dapat menjadi sebagai variabel basis diakhir perhitungan, melainkan harus dikategorikan sebagai variabel non basis. Dengan demikian, dibutuhkan suatu prosedur untuk memindahkan *artificial variables* menjadi variabel non basis. Prosedur ini dilakukan dengan *artificial objective function.*

Artificial objective function ditentukan dengan menjumlahkan seluruh *artificial variables* pada fungsi kendala yang ada. Dimana *artificial objective function* diharapkan pada akhir perhitungan akan bernilai 0, yang mengindikasikan bahwa semua *artificial variables* telah keluar dari solusi basis. Pada kondisi (w=0) model hanya mengandung variabel dasar yang dapat dikontrol. *Artificial objective function* tersebut dikategorikan sebagai fase pertama dari *simplex* dua fase. Setelah objektif ini terpenuhi, baru dapat dilakukan fase 2 berupa pencarian kondisi optimal seperti syarat dan langkah pada *simplex* satu fase. Selain itu, pada fase pertama *simplex* ini indikator optimasi yang digunakan digambarkan dengan d_1.

$$w = \sum x_i \tag{3.43}$$

Contoh Soal 3.2

Kementerian terkait sedang diminta untuk melakukan rasionalisasi tarif untuk angkutan mobil pribadi dan angkutan barang pada sebuah rute penyeberangan. Rute tersebut dikeluhkan operator sebagai rute yang merugikan. Rute tersebut memang pada awalnya dimaksudkan untuk meningkatkan perdagangan antar daerah, sehingga muatan kapal didominasi oleh angkutan barang (i.e., 40 slot) dan sedikit angkutan pribadi (i.e., 5 slot). Selain itu, tarif angkutan barang jauh lebih murah dibandingkan angkutan pribadi, yaitu Rp. 100.000 untuk angkutan barang, dan Rp. 300.000 untuk angkutan pribadi. Asumsikan jumlah yang tersedia tetap. Oleh karenanya diperlukan rasionalisasi terkait tarif ini, dengan beberapa pertimbangan berikut, yaitu:

- Pendapatan dari tarif harus sama dengan atau lebih besar dari biaya operasional kapal selain asuransi (i.e., 8 juta rupiah per perjalanan).
- Adanya biaya asuransi yang ditetapkan sebesar 1% dari tarif untuk angkutan pribadi dan 2 % untuk angkutan barang. dengan total kenaikan asuransi akibat rasionalisasi tidak lebih dari Rp. 10.000
- Faktor perubahan tarif angkutan barang harus lebih besar dibandingkan angkutan pribadi terkait tonase dan ruang yang digunakan

Atas dasar permasalahan tersebut, dapat diidentifikasi fitur utama optimasi yaitu:

- Variabel keputusan yang harus diambil adalah berapa besar (faktor) perubahan tarif untuk angkutan pribadi (x_1) dan angkutan barang (x_2).
- Fungsi tujuan dapat diambil sebagai meminimalkan perubahan tarif pada angkutan barang agar biaya angkutan barang diupayakan serendah mungkin, dapat dilihat pada Persamaan (3.44).
- Fungsi kendala yang terdiri dari beberapa batasan, yaitu
  - Pendapatan dari tarif harus sama dengan atau lebih besar dari biaya operasional kapal selain asuransi dapat dituliskan pada Persamaan (3.45). Dengan mengganggap slot tetap maka kendala ini dapat ditulis:

$$(300.000)(5)x_1 + (100.000)(40)x_2 \geq 8.000.000$$
$$(1.500.000)x_1 + (4.000.000)x_2 \geq 8.000.000$$
$$1{,}5x_1 + 4x_2 \geq 8$$

  - Perubahan asuransi akibat rasionalisasi tarif maksimal 10.000, dapat dituliskan pada Persamaan (3.46).

$$(300.000)(1\%)x_1 + (100.000)(2\%)x_2 \leq 10.000$$
$$3.000x_1 + 2.000x_2 \leq 10.000$$
$$3x_1 + 2x_2 \leq 10$$

  - Faktor perubahan tarif angkutan barang harus lebih besar dibandingkan angkutan pribadi, dapat dituliskan pada Persamaan (3.47).

$$x_1 \leq x_2$$

  - Faktor perubahan tarif tidak boleh negatif, dapat dituliskan pada Persamaan (3.44).

Sehingga dengan menggunakan informasi diatas, dapat disusun model matematika berikut:

$$Min\ Z = x_2 \tag{3.44}$$

$$1{,}5x_1 + 4x_2 \geq 8 \tag{3.45}$$

$$3x_1 + 2x_2 \leq 10 \tag{3.46}$$

$$x_1 - x_2 \leq 0 \tag{3.47}$$

$$x_1 \geq 0, x_2 \geq 0 \tag{3.48}$$

Agar dapat diselesaikan dengan *simplex*, maka perlu dipastikan terlebih dahulu bahwa bentuk *objective function* adalah meminimalkan masalah, tanda di sebelah kanan sudah positif, dan terdapat *non negativity constraint.* Selanjutnya *constraint* akan ditransformasi menjadi persamaan dengan menambahkan *surplus variable* x_3 untuk *constraint* pertama, *slack variable* x_4 untuk *constraint* kedua, dan *slack variable* x_5 untuk *constraint* ketiga. Karena pada *constraint* pertama terdapat *surplus variable* yang bernilai negatif, maka pada persamaan tersebut juga perlu ditambahkan *artificial variable* x_6 pada *constraint* pertama. Dengan demikian, didapatkan persamaan berikut.

$$Min\ Z = x_2 \tag{3.49}$$

$$1{,}5x_1 + 4x_2 - x_3 + x_6 = 8 \tag{3.50}$$

$$3x_1 + 2x_2 + x_4 = 10 \tag{3.51}$$

$$x_1 - x_2 + x_5 = 0 \tag{3.52}$$

$$x_1, x_2, x_3, x_4, x_5, x_6 \geq 0 \tag{3.53}$$

Karena terdapat *artificial variable,* maka dibutuhkan *artificial objective function* yang merupakan penjumlahan dari *artificial variable,* yang ditulis sebagai berikut.

$$w = \sum x_i \tag{3.54}$$

$$w = x_6 = 8 - (1{,}5x_1 + 4x_2 - x_3) \tag{3.55}$$

$$w - 8 = -1{,}5x_1 - 4x_2 + x_3 \tag{3.56}$$

Karena fungsi kendala selain *non-negativity* berjumlah 3, maka jumlah basis adalah 3 dengan solusi basis inisial ditentukan x_6, x_4, dan x_5. Dengan demikian, dapat dibuat

augmented matrix dari dua *objective function* dan ketiga persamaan *constraint* di atas, seperti yang terlihat pada Tabel 3.5.

Tabel 3.5 Matriks iterasi tahap 1 soal 3.2.

x_B	x_1	x_2	x_3	x_4	x_5	x_6	b
x_6	1,5	4	-1	0	0	1	8
x_4	3	2	0	1	0	0	10
x_5	1	-1	0	0	1	0	0
c_l	0	1	0	0	0	0	z-0
d_l	-1,5	-4	1	0	0	0	w-8

Tahap pertama yang dilakukan adalah dengan menjadikan nilai w=0. Langkah yang dilakukan sama seperti pada *simplex* satu fase, yaitu dengan menentukan *pivot column* dari nilai negatif yang terbesar, yakni -4 berada pada kolom x_2. Sehingga x_2 akan dijadikan solusi basis berikutnya. *Pivot row* ditentukan dengan melihat nilai b dibagi dengan a (koefisien pada *pivot column)*. Nilai b/a yang positif terkecil adalah sebesar 2 pada baris x_6 dengan demikian baris yang akan dikeluarkan pada solusi basis adalah x_6 dan akan digantikan dengan solusi x_2.

Tabel 3.6 Matriks dengan *pivot row* dan *pivot column* tahap 1 soal 3.2.

x_B	x_1	x_2	x_3	x_4	x_5	x_6	b	b/a	
x_6	1,5	4	-1	0	0	1	8	9/4=2	R1
x_4	3	2	0	1	0	0	10	10/2=5	R2
x_5	1	-1	0	0	1	0	0	0	R3
c_l	0	1	0	0	0	0	z-0		R4
d_l	-1,5	-4	1	0	0	0	w-8		R5

Keterangan: *highlight* merah: *pivot row*, *kuning*: *pivot column*, hijau: *pivot number*.

Proses selanjutnya adalah dengan membagi *pivot row* dengan *pivot number* dan nilai pada *pivot column* selain *pivot number* harus dibuat menjadi 0 dengan mengoperasikan setiap baris terhadap *pivot row* yang baru. Pengoperasian tersebut dapat ditulis sebagai berikut.

$$R1' = \frac{R1}{4} \tag{3.57}$$

$$R2' = R2 - R1' \times 2 \tag{3.58}$$

$$R3' = R3 + R1' \times 1 \tag{3.59}$$

$$R4' = R4 - R1' \times 1 \tag{3.60}$$

$$R5' = R5 + R1' \times 4 \tag{3.61}$$

Dengan pengoperasian tersebut, maka bisa didapatkan *augmented matrix* yang baru, yang dapat dilihat pada tabel berikut.

Tabel 3.7 Matriks iterasi tahap 2 (soal 3.2.)

x_B	x_1	x_2	x_3	x_4	x_5	x_6	b	
x_2	0,38	1,00	-0,25	0,00	0,00	0,25	2,00	R1’
x_4	2.25	0,00	0,50	1,.00	0,00	-0,50	6,00	R2’
x_5	1,38	0,00	-0,25	0,00	1,00	0,25	2,00	R3’
c_l	-0,38	0,00	0,25	0,00	0,00	-0,25	-2,00	R4’
d_l	3,00	0,00	0,00	0,00	0,00	1,00	w-0	R5’

Pada baris d_l, nilai w sudah sama dengan 0, *artificial variable* sudah tidak termasuk dalam solusi basis, sehingga *objective function* untuk *artificial variable* sudah terpenuhi. Untuk langkah selanjutnya baris d_l dan kolom x_6 sudah dapat dihilangkan dan analisis terfokus untuk menyelesaikan permasalahan *objective function* agar c_l bernilai positif.

Tahapan selanjutnya dilakukan dengan langkah yang sama, yaitu dengan menentukan *pivot column* dari nilai negatif yang terbesar, yakni -0,38 pada kolom x_1, sehingga x_1 akan dijadikan solusi basis berikutnya. *Pivot row* ditentukan dengan melihat nilai b dibagi dengan a (koefisien pada *pivot column)* yang dapat dilihat pada tabel di bawah. Nilai b/a yang positif terkecil adalah sebesar 0,18 pada baris x_5, dengan demikian baris yang akan dikeluarkan pada solusi basis adalah x_5 dan akan digantikan dengann solusi x_1.

Tabel 3.8 Matriks dengan *pivot row* dan *pivot column* tahap 2 (soal 3.2).

x_B	x_1	x_2	x_3	x_4	x_5	b	b/a	Ket
x_2	0.38	1,00	-0,25	0,00	0,00	0,25	0,67	R1
x_4	2,25	0,00	0,50	1,00	0,00	-0,50	-0,22	R2
x_5	1,38	0,00	-0,25	0,00	1,00	0,25	0,18	R3
c_l	-0,38	0,00	0,25	0,00	0,00	0,25		

Keterangan: *highlight* merah: *pivot row*, *kuning*: *pivot column*, hijau: *pivot number*.

Proses selanjutnya adalah dengan membagi *pivot row* dengan *pivot number* dan nilai pada *pivot column* selain *pivot number* harus dibuat menjadi 0 dengan mengoperasikan setiap baris terhadap *pivot row* yang baru. Pengoperasian tersebut dapat ditulis sebagai berikut.

$$R1' = R1 - R3' \times 0{,}38 \tag{3.62}$$

$$R2' = R2 - R3' \times 2{,}25 \tag{3.63}$$

$$R3' = \frac{R3}{1{,}38} \tag{3.64}$$

$$R4' = R4 + R1' \times 0{,}38 \tag{3.65}$$

Dengan pengoperasian tersebut, maka bisa didapatkan *augmented matrix* yang baru, yang dapat dilihat pada tabel berikut.

Tabel 3.9 Matriks dengan evaluasi c_l (soal 3.2).

x_B	x_1	x_2	x_3	x_4	x_5	b	
x_2	0,0	1,0	-0,2	0,0	-0,3	2,2	R1'
x_4	0,0	0,0	0,9	1,0	-1,6	3,1	R2'
x_1	1,0	0,0	-0,2	0,0	0,7	0,2	R3'
c_l	0,0	0,0	0,2	0,0	0,3	Z-2,2	

Pada hasil *augmented matrix* di atas, nilai c_l seluruhnya sudah bernilai positif, sehingga dapat disimpulkan bahwa solusi sudah optimum. Dengan demikian, solusi basis pada permasalahan di atas adalah x_2 dengan nilai 2,2, x_4 dengan nilai 3,1, dan $x_1$dengan nilai 0,2. Oleh karena itu, solusi dari permasalahan di atas adalah $x_1 = 0{,}2$, $x_2 = 2{,}2$, $x_4 = 3{,}1$ dengan $Z = 2{,}2$.

Dari hasil ini dapat diketahui bahwa akan ada peningkatan tarif angkutan barang sebesar 2,2 kali yaitu menjadi Rp. 220.000 per perjalanan. Kondisi ini dipengaruhi oleh kenyataan bahwa tarif dasar angkutan barang sangat rendah jika dibandingkan dengan tarif angkutan pribadi. Sementara tarif angkutan pribadi yang semula adalah Rp. 300.000 dapat turun menjadi Rp. 60.000. Rasionalisasi tarif ini diharapkan tetap mampu menutup operasional dari kapal, dimana dengan tarif ini dengan asumsi slot yang disediakan terisi semua, maka pendapatan dari tarif adalah Rp. 9.100.000

Pembaca yang tertarik untuk melihat penjelasan dalam bentuk audio-visual terkait Bab ini dapat mengunjungi playlist youtube berikut:

https://bit.ly/Playlist-LPSimplex

4. Post Analysis Linear programming

4.1. Pengantar *Post Analysis*

Post analysis bertujuan untuk menganalisis perubahan dari solusi optimum yang sudah diperoleh sebelumnya, jika terjadi perubahan pada koefisien pada fungsi tujuan (i.e., *unit cost*) maupun pada *constraints*. Cara termudah untuk menganalisis perubahan tersebut adalah dengan melakukan iterasi ulang metode *simplex* yang mengakomodasi setiap perubahan-perubahan tersebut. Namun, cara tersebut tidak efisien dan membutuhkan waktu yang lama, terutama jika menghadapi permasalahan optimasi yang besar. Dengan demikian, dibutuhkan suatu metode *post analysis* untuk mengatasi permasalahan tersebut, Secara spesifik buku ini akan membahas *post analysis* dengan mengkondiserasikan perubahan pada koefisien fungsi objektif (i.e., *unit cost c*) dan perubahan pada *resource vektor b*.

4.2. Perubahan Koefisien pada Fungsi Objektif

Perubahan pada *unit cost* pada suatu solusi optimum dapat mengubah i) nilai *fitness*, ii) lokasi solusi optimum, ataupun iii) keduanya. Jika terjadi perubahan *unit cost* secara *uniform,* maka tidak akan terjadi rotasi pada *isoline,* sehingga lokasi optimal tidak berubah akan tetapi terjadi perubahan pada nilai *fitness* saja. Pada kondisi *original, linear programming* dapat ditulis pada Persamaan (4.1)–(4.3).

$$Min\ z = c'x \tag{4.1}$$

$$Ax = b \tag{4.2}$$

$$x \geq 0 \tag{4.3}$$

Jika terjadi perubahan pada *unit cost,* maka terjadi perubahan dari c' menjadi $\underline{c'}$. Salah satu cara yang cukup efisien dalam menganalisis perubahan *unit cost* ini adalah dengan menggunakan hasil terakhir dari iterasi *simplex* pada permasalahan *original*. Penggunaan iterasi terakhir sebagai *objective function* baru akibat perubahan *unit cost*, diharapkan dapat meminimalkan waktu komputasi. Untuk mengimplementasikan prosedur ini, berikut beberapa tahapan yang perlu dilakukan.

1. Mengasumsikan solusi optimum x^* yang didapatkan dari *original linear programming* dan menghitung z^0.
2. Mendefinisikan *indicator lines* yang baru, dengan menggunakan *objective function* yang baru, yakni $\underline{c'}$.

3. Melakukan uji optimal dari solusi pada *objective function* $\underline{c}'$ yang baru. Jika seluruh elemen pada $\underline{c}'$ sudah positif, maka solusi optimum x^* tidak berubah. Namun, jika masih terdapat elemen pada $\underline{c}'$ yang negatif, maka solusi optimum perlu dicari kembali dengan *simplex method.*

Sebelum membahas lebih jauh terkait prosedur untuk melakukan *post analysis* karena perubahan *unit cost* , hubungan fundamental dari *unit cost* ini dibahas terlebih dahulu. Hubungan tersebut dapat dilihat pada Persamaan (4.4).

$$c_l^T = c_N^T - c_B^T A_B^{-1} A_N^* \tag{4.4}$$

Adanya perubahan pada *unit cost* menyebabkan perubahan pada persamaan di atas untuk c_l^T, c_N^T, dan c_B^T. Ada pun diketahui bahwa A_B^{-1} merupakan matriks identitas yang bernilai 1 atau I. Dengan demikian setelah adanya perubahan pada *unit cost,* persamaan tersebut menjadi:

$$\underline{c}_l^T = \underline{c}_N^T - \underline{c}_B^T A_B^{-1^*} A_N^* \tag{4.5}$$

$$\underline{c}_l^T = \underline{c}_N^T - \underline{c}_B^T A_N^* \tag{4.6}$$

Objective function untuk z^k sebelum terjadi modifikasi dapat ditulis sebagai berikut.

$$z^k = c_B^T A_B^{-1^*} b^* \tag{4.7}$$

Dengan adanya modifikasi, maka akan terdapat perubahan untuk $\underline{c}_l^T$, $\underline{c}_N^T$, dan $\underline{c}_B^T$ dan *fitness value* a_0 yang akan menggantikan z^k. Dengan demikian, *objective function* untuk *unit cost* yang telah dimodifikasi dapat ditulis sebagai berikut.

$$c_B^T \rightarrow \underline{c}_B^T \; ; \; z^k \rightarrow a_0 \tag{4.8}$$

$$a_0 = \underline{c}_B^T A_B^{-1^*} b^* \tag{4.9}$$

$$a_0 = \underline{c}_B^T b^* \tag{4.10}$$

Dengan demikian, model baru dari *simplex* setelah adanya modifikasi dapat dituliskan pada matriks berikut (Persamaan (4.11)). Untuk lebih memahami proses dari *post analysis* akibat perubahan *unit cost* maka prosedur penyelesaiannya akan dibahas pada **Contoh Soal 4.1**

$$\begin{array}{cccccc} x_B^* & A_N^* & I & b^* & b/a \\ - & c_l' & 0' & (z - a^0) & - \end{array} \tag{4.11}$$

Contoh Soal 4.1

Sebuah lembaga bermaksud untuk memaksimalkan *eksposure* paket bantuannya kepada masyarakat terdampak bencana. Anggaplah dalam keadaan bencana, diketahui bahwa terdapat dua buah lokasi pengungsian yang perlu mendapatkan bantuan. Jumlah masyarakat terdampak dianggap pada lokasi dua adalah tiga kali penduduk lokasi satu. Selain itu, diketahui bahwa hanya tersedia empat buah jenis kendaraan dengan kapasitas muatan yang sama. Selain itu, pada lokasi satu karena terdapat pembatasan dari pemerintah, jumlah kendaraan yang dapat masuk adalah maksimal tiga kendaraan. Atas dasar permasalahan ini,

a. Bagaimana solusi optimum pada permasalahan di atas menggunakan metode *simplex*?
b. Bagaimana solusi optimum jika nilai c_2 berubah menjadi 0 (anggap daerah dua tidak terkena bencana)?
c. Bagaimana solusi optimum jika nilai c_1 berubah menjadi 4 (anggap lokasi 1 memiliki dampak keparahan yang lebih tinggi)?

Berdasarkan kondisi ini dapat dirumuskan permasalahan optimasi sebagai berikut;
x_1: adalah jumlah kendaraan bantuan yang akan dikirimkan kepada lokasi pengungsian 1
x_2: adalah jumlah kendaraan bantuan yang akan dikirimkan kepada lokasi pengungsian 2

Sehingga fungsi tujuan dapat dianggap sebagai perkalian jumlah penduduk yang terekspos bantuan dengan jumlah bantuan yang dikirim (i.e., jumlah kendaraan yang dikirimkan). Fungsi kendala 1 dan 2 masing-masing menujukan batasan dari sisi jumlah kendaraan yang tersedia, dan batasan kendaraan untuk menuju lokasi satu.

$$Max\ z = x_1 + 3x_2 \tag{4.1}$$

$$x_1 + x_2 \leq 4 \tag{4.2}$$

$$x_1 \leq 3 \tag{4.3}$$

$$x_1 \geq 0; x_2 \geq 0 \tag{4.4}$$

a. Solusi optimum menggunakan metode *simplex*.

Agar permasalahan di atas dapat diselesaikan dengan *simplex*, maka perlu dipastikan terlebih dahulu bahwa bentuk *objective function* adalah meminimalkan masalah, maka tanda pada *objective function* pun berubah menjadi negatif untuk menyesuaikan dengan permasalahan minimasi. Selain itu, perlu dipastikan tanda di sebelah kanan sudah positif dan terdapat *non negativity constraint.* Selanjutnya pertidaksamaan pada *constraint* akan ditransformasi menjadi bentuk persamaan dengan menambahkan *slack variable* x_3 untuk

constraint pertama, dan *slack variable* x_4 untuk *constraint* kedua. Dengan demikian, didapatkan tiga persamaan berikut. Transformasi Persamaan (4.5) – (4.7) menjadi *augmented matrix* dapat dilihat pada Tabel 4.1

$$Min\ z = -x_1 - 3x_2 \tag{4.5}$$

$$x_1 + x_2 + x_3 = 4 \tag{4.6}$$

$$x_1 + x_4 = 3 \tag{4.7}$$

Tabel 4.1 Matriks beserta *pivot column* dan *pivot row.*

x_b	x_1	x_2	x_3	x_4	b	b/a	
x_3	1	1	1	0	4	4	*R1*
x_4	1	0	0	1	3	*inf*	*R2*
c_l	-1	-3	0	0	0		

Keterangan: *highlight* merah: *pivot row*, *kuning*: *pivot column*, hijau: *pivot number*.

Proses selanjutnya adalah dengan membagi *pivot row* dengan *pivot number* dan nilai pada *pivot column* selain *pivot number* harus dibuat menjadi 0 dengan mengoperasikan setiap baris terhadap *pivot row* yang baru. Pengoperasian tersebut dapat ditulis sebagai berikut.

$$R1' = R1 \tag{4.8}$$

$$R2' = R2 \tag{4.9}$$

$$R3' = R3 + 3\,R1 \tag{4.10}$$

Dengan pengoperasian tersebut, maka bisa didapatkan *augmented matrix* yang baru, yang dapat dilihat pada Tabel 4.2.

Tabel 4.2 Matriks hasil metode *simplex* (soal 4.1a)

x_b	x_1	x_2	x_3	x_4	b	
x_2	1	1	1	0	4	*R1'*
x_4	1	0	0	1	3	*R2'*
c_l	2	0	3	0	*z*-12	

Dari hasil tabel tersebut, didapatkan bahwa nilai pada baris c_l sudah bernilai positif maka bisa didapatkan solusi optimum, yakni $x_1^* = 0, x_2^* = 4, x_3^* = 0, x_4^* = 3$ dengan *fitness value* $z^* = 12, a^* = -12$. Artinya untuk meningkatkan *eksposure* bantuan lembaga tersebut penting untuk mengirimkan keempat kendaraannya kepada lokasi dua.

b. Solusi optimum jika nilai c_2 berubah menjadi 0.

Pada kasus modifikasi ini, maka nilai *c* untuk setiap variabel dapat ditulis sebagai berikut.

$$[c_1 \quad c_2 \quad c_3 \quad c_4] = [-1 \quad 0 \quad 0 \quad 0] \tag{4.11}$$

Dengan hasil iterasi terakhir pada hasil solusi optimum sebelumnya, maka solusi basis adalah pada x_2 dan x_4 dan solusi non basis adalah pada x_1 dan x_3, dapat ditulis pada persamaan berikut.

$$x_B = \begin{bmatrix} x_2 \\ x_4 \end{bmatrix}, x_N = \begin{bmatrix} x_1 \\ x_3 \end{bmatrix} \tag{4.12}$$

$$c_B = [c_2 \quad c_4] = [0 \quad 0] \tag{4.13}$$

$$c_N = [c_1 \quad c_3] = [-1 \quad 0] \tag{4.14}$$

Sedangkan, matriks A_N^* dapat ditentukan dari tabel hasil iterasi akhir yang merupakan nilai-nilai koefisien di bawah variabel non basis (x_1 dan x_3).

$$A_N^* = \begin{bmatrix} a_1^2 & a_3^2 \\ a_1^4 & a_3^4 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \tag{4.15}$$

Tahapan selanjutnya adalah dengan menghitung $\underline{c_l}^T$ yang dimodifikasi, yakni sebagai berikut.

$$\underline{c_l}^T = \underline{c_N}^T - \underline{c_B}^T A_B^{-1*} A_N^* \tag{4.16}$$

$$\underline{c_l^T} = [-1 \quad 0] - [0 \quad 0] \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \tag{4.17}$$

$$\underline{c_l^T} = [-1 \quad 0] \tag{4.18}$$

Nilai a_0 juga dapat dihitung dengan persamaan berikut.

$$a_0 = \underline{c_B'} b^* = [0 \quad 0] \begin{bmatrix} 4 \\ 3 \end{bmatrix} = [0 \quad 0] \tag{4.19}$$

Hasil perhitungan di atas bisa dituliskan pada tabel *simplex* di bawah. Nilai koefisien pada baris c_l ditulis berdasarkan koefisien non basis pada Persamaan (4.18) dan nilai z diambil dari nilai a_0.

Tabel 4.3 Input informasi pada matriks di iterasi akhir (soal 4.1b).

x_b	x_1	x_2	x_3	x_4	b	*Ket.*
x_2	1	1	1	0	4	R1'
x_4	1	0	0	1	3	R2'
c_l	-1	0	0	0	Z-0	

Diinputkan pada matrik di iterasi terakhir

$\underline{c_l}^T = [\underline{c_1} \quad \underline{c_3}] = [-1 \quad 0]$

$a^0 = \underline{c_B}' b^* = 0$

Berdasarkan hasil tersebut, maka disimpulkan bahwa solusi tersebut belum optimal karena masih terdapat nilai koefisien yang negatif pada c_l. Dengan demikian, dibutuhkan iterasi dengan *simplex* kembali agar nilai c_l positif. *Pivot column* ditentukan pada kolom x_1 dan *pivot row* ditentukan pada baris x_4 dan didapatkan hasil pada tabel berikut.

Tabel 4.4 Matriks hasil metode *simplex* (soal 4.1b)

x_b	x_1	x_2	x_3	x_4	b
x_2	0	1	1	-1	1
x_1	1	0	0	1	3
c_l	0	0	0	1	z+3

Hasil iterasi yang terlihat pada Tabel 4.4 sudah didapatkan nilai positif pada seluruh koefisien pada c_l. Sehingga solusi optimum adalah $x_1^* = 3, x_2^* = 1, x_3^* = 0, x_4^* = 0$ dengan *fitness value* $z^* = -3, a^* = 3$. Pada kondisi adanya perubahan, maka lembaga tersebut harus mengirimkan bantuannya kepada lokasi satu sebanyak satu kendaraan. Meskipun jumlah kendaraan yang tersedia adalah empat kendaraan, lembaga tersebut tidak dapat mengirimkan keempatnya akibat adanya batasan jumlah kendaraan untuk memasuki lokasi satu. Selain itu, *simplex* masih memberikan solusi optimal pada lokasi dua yaitu sebanyak 1 kendaraan walaupun daerah dua tidak terdampak bencana. Kondisi ini disebabkan oleh masih dimasukkannya lokasi dua dalam fungsi objektif tanpa menghilangkannya pada permasalahan *simplex*, sehingga x_2 masih dimasukkan selama proses iterasi *simplex*.

c. Solusi optimum jika nilai c_1 berubah menjadi 4.

Pada kasus modifikasi ini, maka nilai *c* untuk setiap variabel dapat ditulis sebagai berikut.

$$[c_1 \quad c_2 \quad c_3 \quad c_4] = [-4 \quad -3 \quad 0 \quad 0] \tag{4.20}$$

Dengan hasil iterasi terakhir hasil solusi optimum sebelumnya pada **soal a**, maka solusi basis adalah pada x_2 dan x_4 dan solusi non basis adalah pada x_1 dan x_3, dapat ditulis pada persamaan berikut.

$$x_B = \begin{bmatrix} x_2 \\ x_4 \end{bmatrix}, x_N = \begin{bmatrix} x_1 \\ x_3 \end{bmatrix} \tag{4.21}$$

$$c_B = [c_2 \quad c_4] = [-3 \quad 0] \tag{4.22}$$

$$c_N = [c_1 \quad c_3] = [-4 \quad 0] \tag{4.23}$$

Sedangkan, matriks A_N^* dapat ditentukan dari tabel hasil iterasi akhir yang merupakan nilai-nilai koefisien di bawah variabel non basis ($x_1 \; dan \; x_3$).

$$A_N^* = \begin{bmatrix} a_1^2 & a_3^2 \\ a_1^4 & a_3^4 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \tag{4.24}$$

Tahapan selanjutnya adalah dengan menghitung c_l^T yang dimodifikasi, yakni sebagai berikut.

$$\underline{c}_l^T = \underline{c}_{\underline{N}}^T - \underline{c}_{\underline{B}}^T A_B^{-1*} A_N^* \tag{4.25}$$

$$\underline{c_l}^T = [-4 \quad 0] - [-3 \quad 0] \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \tag{4.26}$$

$$\underline{c_l}^T = [-1 \quad 3] \tag{4.27}$$

Nilai a_0 juga dapat dihitung dengan persamaan berikut.

$$a_0 = \underline{c_B'} b^* = [-3 \quad 0] \begin{bmatrix} 4 \\ 3 \end{bmatrix} = -12 \tag{4.28}$$

Hasil perhitungan di atas bisa dituliskan pada tabel *simplex* di bawah. Nilai koefisien pada baris c_l ditulis berdasarkan koefisien non basis pada Persamaan (4.27). dan nilai z diambil dari nilai a_0.

Tabel 4.5 Matriks input informasi pada matriks di iterasi akhir (soal 4.1c).

x_b	x_1	x_2	x_3	x_4	b	*Ket.*
x_2	1	1	1	0	4	R1’
x_4	1	0	0	1	3	R2’
c_l	-1	3	0	0	Z+12	

Diinputkan pada matrik di iterasi terakhir

$\underline{c}_l^T = [\underline{c}_1 \quad \underline{c}_3] = [-1 \quad 3]$

$a^0 = \underline{c_B}'b^* = -12$

Berdasarkan hasil tersebut, maka disimpulkan bahwa solusi tersebut belum optimal karena masih terdapat nilai koefisien yang negatif pada c_l. Dengan demikian, dibutuhkan iterasi dengan *simplex* kembali agar nilai c_l positif. *Pivot column* ditentukan pada kolom x_1 dan *pivot row* ditentukan pada baris x_4 dan didapatkan hasil pada tabel berikut.

Tabel 4.6 Matriks hasil metode *simplex* (soal 4.1c).

x_b	x_1	x_2	x_3	x_4	b
x_2	0	1	1	-1	1
x_1	1	0	0	1	3
c_l	0	3	0	1	z+15

Dari tabel diatas terlihat bahwa telah didapatkan nilai positif pada seluruh koefisien pada c_l. Sehingga solusi optimum adalah $x_1^* = 3, x_2^* = 1, x_3^* = 0, x_4^* = 0$ dengan *fitness value* $z^* = -15, a^* = 15$. Nilai ini memberikan ilustrasi ketika keparahan di lokasi meningkat (jumlah korban lebih banyak) maka untuk meningkatkan *eksposure* bantuan, lembaga tersebut dapat mengirimkan kendaraan lebih banyak ke lokasi 1 dibandingkan ke lokasi 2.

4.3. Shadow Prices dan Reduced Cost

Istilah *shadow prices and reduced cost* umum digunakan dalam *post analysis* dalam rangka memudahkan pemahaman akan dampak dari perubahan-perubahan pada fungsi kendala kepada solusi optimum. *Shadow prices* dapat didefinisikan sebagai rasio perubahan nilai *fitness* pada solusi optimum (i.e., Δz) terhadap perubahan alokasi *resource* pada *constraint* (i.e., Δb_i). Bentuk persamaan dari *shadow prices* dapat dilihat pada Persamaan (4.29).

$$MC_i = \frac{\Delta z}{\Delta b_i} \tag{4.29}$$

Besar nilai perubahan pada *shadow price* tersebut dapat diambil pada nilai dari hasil iterasi *simplex* yang terakhir. Secara spesifik nilai tersebut berada pada baris c_l di bawah kolom *slack* atau *surplus variable* (lihat Gambar 4.1). Untuk setiap penambahan suatu nilai pada alokasi di fungsi *constraint* tersebut, akan menyebabkan pertambahan *fitness value* sebesar *shadow price*.

x_b	x_1	x_2	x_3	x_4	b
x_2	1	1	1	0	4
x_4	1	0	0	1	3
c_l	1	0	2	0	Z+15
	Reduce Cost	*Reduce Cost*	*Shadow prices*	*Shadow prices*	

Gambar 4.1 Ilustrasi lokasi *shadow price* dan *reduce cost* dari iterasi terakhir table *simplex.*

Sebagai contoh pada Gambar 4.1, maka setiap penambahan alokasi pada fungsi kendala yang terkait x_3 akan menyebabkan penambahan nilai *fitness* sebesar dua kali. Pada Tabel 4.7 terlihat bahwa perubahan sisi kanan (i.e., *resources*/alokasi) fungsi kendala yang terkait dengan x_3 menyebabkan adanya perubahan nilai pada *objective function* (i.e., *fitness value*). Nilai *fitness* bertambah hingga dua kali dengan adanya penambahan *resources* pada fungsi kendala sebanyak satu unit, dan berlaku sebaliknya.

Tabel 4.7 Ilustrasi *shadow price.*

Fungsi Kendala	Solusi Optimal	*Fitness value*
$x_1 + x_2 \leq 2$	x_1 =0, x_2=2	4
$x_1 + x_2 \leq 1$ (*resources* berkurang ½ kalinya)	x_1=0, x_2=1	2
$x_1 + x_2 \leq 4$ (*resources* naik 2 kalinya)	x_1=0, x_2=4	8

Sementara itu, r*educed cost* dapat didefinisikan sebagai perubahan dari *fitness value* dari solusi optimum terhadap perubahan pada *non negativity constraint* untuk setiap variabel primal yang digunakan. Besar nilai perubahan pada *reduced cost* tersebut dapat diambil pada nilai dari hasil iterasi *simplex* yang terakhir pada baris c_l di bawah kolom *primal variable* (lihat Gambar 4.1). Untuk setiap penambahan suatu nilai dari *non negativity constraint,* maka akan mengurangi *fitness value* sebesar *reduced cost* yang didapatkan tersebut. Sebagai contoh, Tabel 4.8 memberikan ilustrasi perubahan *non-negativity constraint* x_1 menjadi $x_1 \geq 1$, yang menyebabkan perubahan nilai *fitness* dari 4 menjadi 3, sesuai dengan nilai *reduced cost* x_1 yang sama dengan 1. Sementara itu, perubahan pada *non-negativity constraint* x_2 tidak menyebabkan perubahan pada *fitness value*, karena nilai *reduced cost*-nya yang sama dengan 0. Kondisi *reduce cost* yang sama dengan nol terjadi pada variabel yang merupakan variabel basis dari solusi terakhir *simplex*. *Reduced cost* juga dapat dianggap sebagai *opportunity cost*, karena adanya perubahan *non-negativity constraint* menyebabkan adanya potensi kehilangan dari sisi *fitness value*.

Tabel 4.8 Ilustrasi *reduced cost.*

No	Fungsi Kendala	Solusi Optimal	*Fitness value*
1	$x_1 \geq 0$; $x_2 \geq 0$	$x_1 = 0$, $x_2 = 2$	4
2	$x_1 \geq 1$; $x_2 \geq 0$ (kendala non negatif x_1 berubah menjadi 1)	$x_1 = 1$, $x_2 = 1$	3
3	$x_1 \geq 0$; $x_2 \geq 1$ (kendala non negatif x_2 berubah menjadi 1)	$x_1 = 0$, $x_2 = 2$	4

Namun demikian, penggunaan nilai *shadow price* dan *reduced cost* hanya dapat digunakan untuk mengestimasi perubahan yang bernilai kecil pada *constraint,* bergantung pada batas bawah dan atas pada perubahan tersebut. Untuk perubahan yang besar, maka estimasi dari hasil *fitness value* juga menjadi kurang akurat dalam menentukan lokasi titik optimum yang baru. Contoh soal 4.2 memberikan ilustrasi penggunaan *shadow price* dan *reduced cost*.

Contoh Soal 4.2

Operator Pelabuhan bermaksud memaksimalkan pendapatan lapangan penumpukan dengan menetapkan ukuran dari luas lapangan kontainer berdasarkan tipenya. Anggap terdapat dua jenis kontainer yang umumnya ada di lapangan penumpukan itu. Jika diketahui bahwa tarif per m^2 untuk kontainer B lebih mahal 3 kali dibandingkan tipe A. Selain itu, luas yang tersedia untuk lapangan penumpukan adalah 4 ribu m^2. Sesuai dengan estimasi kebutuhan masing-masing tipe kontainer, diketahui bahwa tipe A hanya memiliki maksimal 1 ribu m^2 sementara tipe B 5 ribu m^2.

a. Berapa luas dari masing-masing tipe untuk memaksimalkan keuntungan ?
b. Jika total luas meningkat menjadi 5 ribu m^2, berapa luas dari masing tipe untuk memaksimalkan keuntungan?
c. Jika luas lapangan penumpukan tipe B dengan luas awal 5 ribu m^2 diubah menjadi 4 ribu m^2, berapa luas dari masing-masing tipe untuk memaksimalkan keuntungan?
d. Jika lapangan penumpukan dari tipe A harus disediakan, berapa luas dari masing-masing tipe untuk memaksimalkan keuntungan?
e. Jika lapangan penumpukan dari tipe B harus disediakan, berapa luas dari masing-masing tipe untuk memaksimalkan keuntungan?

a. Solusi optimum

Anggaplah variabel keputusan untuk permasalahan optimasi tersebut dimodelkan sebagai berikut:

x_1: adalah luas lapangan penumpukan untuk kontainer tipe A

x_2: adalah luas lapangan penumpukan untuk kontainer tipe B

Permasalahan tersebut kemudian dituliskan menjadi bentuk matematika sebagai berikut yang mencakup *objective function* dan *constraints.*

$$Max\ z = x_1 + 3x_2 \quad (4.30)$$
$$x_1 + x_2 \leq 4 \quad (4.31)$$
$$x_1 \leq 1 \quad (4.32)$$
$$x_2 \leq 5 \quad (4.33)$$
$$x_1 \geq 0, x_2 \geq 0 \quad (4.34)$$

Selanjutnya, akan ditambahkan *slack variable* untuk mengubah pertidaksamaan pada *constraint* menjadi bentuk persamaan.

$$x_1 + x_2 + x_3 = 4 \quad (4.35)$$
$$x_1 + x_4 = 1 \quad (4.36)$$
$$x_2 + x_5 = 5 \quad (4.37)$$

Dengan menggunakan metode *simplex* pada langkah-langkah pada Bab 3 sebelumnya, maka bisa didapatkan hasil matriks iterasi terakhir yang ditunjukkan pada Tabel 4.9.

Tabel 4.9 Matriks hasil metode *simplex* (soal 4.2a).

x_b	x_1	x_2	x_3	x_4	x_5	b
x_2	1	1	1	0	0	4
x_4	1	0	0	1	0	1
x_5	-1	0	-1	0	1	1
c_j	2	0	3	0	0	z+12

Berdasarkan tabel tersebut, maka bisa didapatkan bahwa solusi basis terdapat pada variabel $x_2, x_4, dan\ x_5$. Dengan demikian, solusi optimum pada permasalahan tersebut berada pada $x_1^* = 0, x_2^* = 4$ dengan *fitness value* yang dihasilkan sebesar 12-unit satuan keuntungan.

b. **Solusi optimum jika total luas meningkat menjadi 5 ribu m²**

Pada kondisi ini, maka akan terdapat perubahan alokasi *resource* dari *constraint* pertama pada Persamaan (4.31). Analisis ini dapat dilakukan dengan nilai *shadow price* yang didapatkan dari nilai pada baris c_l dengan kolom *slack variable* x_3 pada *constraint* ini. Nilai yang didapatkan adalah 3 dengan arti bahwa untuk setiap penambahan nilai alokasi *resource* pada Persamaan (4.31), maka terdapat peningkatan *fitness value* sebanyak 3. Namun demikian, penambahan ini juga terbatas untuk batasan penambahan tertentu. Batasan ini bisa didapatkan dengan menggunakan aplikasi *solver* yang terdapat pada *Microsoft Excel,* dapat dilihat pada Gambar 4.2 pada baris pertama. Penggunaan aplikasi *Ms. Excel* akan dibahas lebih lanjut pada Sub Bab 4.4.

Ms. Excel Solver Output

Cell	Name	Final Value	Shadow Price	Constraint R.H. Side	Allowable Increase	Allowable Decrease
$D$5	Persamaan 4.31	4	-3	4	1	4
$D$6	Persamaan 4.32	0	0	1	1E+30	1
$D$7	Persamaan 4.33	4	0	5	1E+30	1

Gambar 4.2 Hasil *sensitivity analysis* dari *solver* untuk *shadow price.*

Berdasarkan tabel tersebut, didapatkan bahwa penambahan *shadow price* pada *constraint* pertama hanya dapat ditambahkan sebanyak satu *grade*. Dengan demikian, untuk perubahan alokasi menjadi 5 masih bisa menggunakan hasil *shadow price,* yakni *fitness value* yang berubah menjadi 15 unit satuan keuntungan dengan solusi optimum yang sama. Hal ini dapat dilihat pula pada Tabel 4.10 yang membuktikan bahwa jika masih dalam rentang *allowable increase* maka pendekatan *shadow price* dapat digunakan. Akan tetapi ketika diluar dari rentangnya maka penggunaan *shadow price* ini menjadi tidak relevan.

Tabel 4.10 Hasil perubahan solusi optimum pada perubahan *constraint 1.*

Fungsi Kendala	Solusi Optimal	*Fitness values*
$x_1 + x_2 \leq 4$	x_1=0, x_2 =4	12
$x_1 + x_2 \leq 5$ (naik 1)	x_1=0, x_2 =5	15
$x_1 + x_2 \leq 7$ (naik 3)	x_1=1, x_2 =5	16

c. Solusi optimum jika luas lapangan penumpukan tipe B dengan luas awal 5 ribu m^2 diubah menjadi 4 ribu m^2

Pada kondisi ini, maka akan terdapat perubahan alokasi *resource* dari *constraint* ketiga pada Persamaan (4.33). Analisis ini dapat dilakukan dengan nilai *shadow price* yang didapatkan dari nilai pada baris c_l dengan kolom *slack variable* x_6 pada *constraint* ini. Nilai yang didapatkan adalah 0 dengan arti bahwa untuk setiap penambahan nilai alokasi *resource* pada Persamaan (4.33), maka tidak terdapat peningkatan atau pengurangan dari *fitness value*. Namun demikian, penambahan ini juga terbatas untuk batasan penambahan tertentu. Batasan ini bisa didapatkan menggunakan aplikasi *solver* yang terdapat pada *Microsoft Excel,* dapat dilihat pada Gambar 4.2. Berdasarkan gambar tersebut, didapatkan bahwa pengurangan *shadow price* pada *constraint* ketiga dapat dilakukan sebanyak 1 *grade*. Dengan demikian, untuk perubahan alokasi menjadi 4 ribu m^2 masih bisa menggunakan hasil *shadow price,* yakni *fitness value* 12 unit satuan keuntungan dengan solusi optimum yang sama.

d. Solusi optimum jika luasan dari tipe A harus disediakan

Jika luasan dari tipe A harus ada yang disewakan, maka terjadi perubahan pada *non negativity constraint* menjadi $x_1 \geq 1, x_2 \geq 0$. Untuk menganalisis kondisi ini, dapat dilakukan dengan melihat nilai *reduced cost* yang didapatkan dari nilai c_l pada kolom *primal variable* x_1 pada iterasi terakhir *simplex.* Pada kolom tersebut, terdapat nilai sebesar 2, dengan demikian, dapat dikatakan bahwa dengan adanya penambahan 1 nilai pada *non negativity constraint* menyebabkan pengurangan dari *fitness value* sebesar 2. Sehingga menjadi untuk kasus ini akan terjadi penurunan *fitness value* dari 12 menjadi 10, dengan solusi optimum menjadi $x_1 = 1, x_2 = 3$ (menggunakan iterasi *simplex*). Rentang dapat digunakan pendekatan *reduced cost* dapat pula diperoleh dengan menggunakan bantuan perangkat lunak *Ms. Excel* yang akan dijelaskan lebih detail pada bab selanjutnya (lihat Gambar 4.3).

Ms. Excel Solver Output

Cell	Name	Final Value	Reduced Cost	Objective Coefficient	Allowable Increase	Allowable Decrease
$B$11	Optimal Sol (DC) x1	0	2	-1	1E+30	2
$C$11	Optimal Sol (DC) x2	4	0	-3	2	1E+30

Gambar 4.3 Hasil *sensitivity analysis* dari *solver* untuk *reduced cost*.

e. Solusi optimum jika lapangan tipe B harus disediakan

Jika luasan dari Tipe B harus disediakan, maka terjadi perubahan pada *non negativity constraint* menjadi $x_1 \geq 0, x_2 \geq 1$. Untuk menganalisis kondisi ini, dapat dilakukan dengan

melihat nilai *reduced cost* yang didapatkan dari nilai c_l pada kolom *primal variable* x_2. Pada kolom tersebut, terdapat nilai sebesar 0, dengan demikian, dapat dikatakan bahwa dengan adanya penambahan 1 nilai pada *non negativity constraint* tidak menyebabkan perubahan dari *fitness value*, yakni 12, dengan posisi solusi optimum yang sama ($x_1^* = 0, x_2^* = 4$).

4.4. Dual Model

Pada pendekatan *linear programming,* sesungguhnya terdapat dua tipe model, yakni *primal model* dan *dual primal model.* Kedua tipe tersebut pada dasarnya memiliki konsep yang sama, namun perspektif yang berbeda. *Primal model* bertujuan untuk memaksimalkan atau meminimalkan suatu fungsi tujuan dari *constraints* yang tersedia, seperti yang sudah dianalisis pada Bab III sebelumnya. Sedangkan, *dual primal model* mampu menganalisis *resources* yang dapat ditingkatkan atau dikurangi penggunaannya agar dapat memaksimalkan atau meminimalkan suatu fungsi tujuan yang ada. Pada kondisi optimum, maka solusi optimum dari *dual model* adalah sama dengan *primal model* (lihat Persamaan (4.49)).

$$d^* = z^* \tag{4.38}$$

Terdapat solusi dari *dual primal model,* maka juga terdapat solusi dari *primal model.* Jika tidak terdapat solusi optimal pada *dual primal model,* maka juga tidak terdapat solusi dari *primal model.*

Model dual umum pula digunakan dalam menganalisis perubahan dari solusi. Secara spesifik analisis *dual model* dilakukan untuk menganalisis solusi optimum jika terjadi suatu perubahan *constraints.* Seperti yang telah dibahas sebelumnya, perubahan pada *binding/active constraints* (i.e., *constraints* yang berada solusi optimum) dapat menyebabkan perubahan pada posisi solusi optimum. Sementara perubahan pada *non-binding/inactive constraints* (i.e., *constraints* yang tidak berada pada solusi optimum), mungkin tidak mengubah posisi dari solusi optimum.

Penerapan model *dual primal* menjadi penting i) untuk memahami interpretasi dari *shadow price* terhadap optimasi *simplex* dan ii) dapat memberikan keunggulan dalam perhitungan yang lebih efisien pada permasalahan optimasi dengan skala yang lebih besar. Prosedur dari penyelesaian *dual model* dapat dilakukan sebagai berikut.

1. Membuat *primal model* dari suatu permasalahan *linear programming*.

$$Min\ z = c'x \quad (4.39)$$
$$Ax \geq b \quad (4.40)$$
$$x \geq 0 \quad (4.41)$$

Membuat *dual model* dengan melakukan transformasi dari *primal model* yang sudah dibuat.

$$Max\ d = b'y \quad (4.42)$$
$$A'y \leq c \quad (4.43)$$
$$y \geq 0 \quad (4.44)$$

2. Memformulasikan *dual model* menjadi bentuk standar dari *linear programming* dengan menambahkan *slack* dan *surplus variable* untuk mentransformasikan pertidaksamaan menjadi bentuk persamaan.

$$Min\ a = b'y \quad (4.45)$$
$$A'y = c \quad (4.46)$$
$$y \geq 0 \quad (4.47)$$

3. Mencari solusi optimum dengan metode *simplex*.
4. Melakukan analisis sensitivitas untuk perubahan pada *resource* dari hasil solusi optimum yang didapatkan.

$$a^o = b'_B\ c^*_B \quad (4.48)$$
$$b_i{}' = [b'_N - b_B{}' A^*_N] \quad (4.49)$$

Untuk memahami lebih jauh terkait aplikasi model dual, **Contoh Soal 4.3** memberikan ilustrasi masalah dan tahapan penyelesaiannya.

Contoh Soal 4.3

Permasalahan pada contoh soal ini ini identik dengan **Contoh Soal 4.2**, meskipun pada contoh ini akan diselesaikan dengan pendekatan *dual*. Adapun beberapa pertanyaan yang perlu dijawab adalah sebagai berikut:

a. Formulasikan *dual model* dan temukan solusi optimum dari permasalahan di atas.
b. Bagaimana perubahan pada solusi optimum jika terdapat perubahan pada *constraint* pertama menjadi 2,5 (i.e., luas lapangan penumpukan dapat menjadi 2,5 rb m^2)?

c. Bagaimana perubahan pada solusi optimum jika terdapat perubahan pada *constraint* ketiga menjadi 1 (i.e., luas lapangan penumpukan tipe B hanya diperbolehkan maksimal 1 rb m^2)?

a. Formulasi *dual model* dan solusi optimum

Hal pertama yang dilakukan adalah memastikan agar bentuk dari permasalahan di atas sudah memenuhi bentuk standar dari *primal model linear programming,* seperti yang dapat dilihat pada Persamaan (4.42). Dengan demikian, bentuk persamaan *primal model* pada permasalahan ini dapat ditulis sebagai berikut untuk menyesuaikan bentuk yang ada.

$$Min\ z = -1x_1 - 3x_2 \tag{4.50}$$

$$-x_1 - x_2 \geq -4 \tag{4.51}$$

$$-x_1 \geq -1 \tag{4.52}$$

$$-x_2 \geq -5 \tag{4.53}$$

$$x_1 \geq 0, x_2 \geq 0 \tag{4.54}$$

Bentuk permasalahan di atas juga dapat dituliskan dalam bentuk matriks sebagai berikut.

$$Min\ z = \begin{bmatrix} -1 & -3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{4.55}$$

$$\begin{bmatrix} -1 & -1 \\ -1 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \geq \begin{bmatrix} -4 \\ -1 \\ -5 \end{bmatrix} \tag{4.56}$$

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \geq \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{4.57}$$

Bentuk dari *primal model* di atas akan ditransformasikan menjadi bentuk standar *dual model* yang dapat dilihat pada Persamaan (4.61) hingga Persamaan (4.63). Pada persamaan *primal model,* maka sudah didapatkan beberapa variabel yang dapat digunakan, yakni c', A, b. Variabel-variabel ini akan disesuaikan pada bentuk standar *dual model*.

$$Max\ d = \begin{bmatrix} -4 & -1 & -5 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} \tag{4.58}$$

$$\begin{bmatrix} -1 & -1 & 0 \\ -1 & 0 & -1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} \leq \begin{bmatrix} -1 \\ -3 \end{bmatrix} \tag{4.59}$$

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} \geq \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{4.60}$$

Selanjutnya, bentuk dari *dual model* tersebut ditransformasikan menjadi bentuk standar *linear programming* yang dapat dilihat sebagai berikut.

$$Min\ d = [4 \quad 1 \quad 5]\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} \tag{4.61}$$

$$\begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} \geq \begin{bmatrix} 1 \\ 3 \end{bmatrix} \tag{4.62}$$

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} \geq \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{4.63}$$

Bentuk dari matriks di atas dapat diubah menjadi bentuk persamaan sebagai berikut.

$$Min\ z = 4y_1 + y_2 + 5y_3 \tag{4.64}$$

$$y_1 + y_2 \geq 1 \tag{4.65}$$

$$y_1 + y_3 \geq 3 \tag{4.66}$$

$$y_1 \geq 0, y_2 \geq 0, y_3 \geq 0 \tag{4.67}$$

Selanjutnya akan dilakukan metode *simplex* untuk menyelesaikan permasalahan di atas. Karena terdapat tanda lebih besar sama dengan pada pertidaksamaan, maka perlu ditambahkan *surplus variable* y_4 dan y_5 serta *artificial variabel* y_6 dan y_7. Dengan ditambahkannya *artificial variable,* maka dilakukan metode *simplex* dua fase yang juga menambah *artificial objective function w.* Dengan demikian, bentuk Persamaan yang akan disolusikan dengan *simplex* dua fase adalah sebagai berikut.

$$Min\ z = 4y_1 + y_2 + 5y_3 \tag{4.68}$$

$$Min\ w = y_6 + y_7 \tag{4.69}$$

$$y_1 + y_2 - y_4 + y_6 = 1 \tag{4.70}$$

$$y_1 + y_3 - y_5 + y_7 = 3 \tag{4.71}$$

Selanjutnya akan dilakukan metode *simplex* dua fase dengan metode yang sama seperti pada bagian sebelumnya. Setelah dilakukan metode *simplex* dua fase, maka bisa didapatkan hasil pada Tabel 4.11. Berbeda dengan pendekatan *primal* hasil dari model *dual* adalah *variabel y* yang mendeskripsikan *shadow price dan reduced cost.* Sementara *decision variable x* dapat dilihat pada bagian bawah dari iterasi terakhir model *dual.* Dari hasil tersebut diperoleh bawah *shadow price* untuk fungsi kendala pertama adalah $y_1^* = 3$, yang

dapat diartikan pada rentang yang diterima, setiap perubahan pada *resource* (b) akan menyebabkan perubahan pada *fitness value* sebesar 3. Sementara itu, perubahan fungsi kendala non negatif x_2 akan menyebabkan perubahan *fitness value* sebesar 2, karena *reduced cost* $y_4^* = 2$.

Tabel 4.11 Ilustrasi matriks akhir metode *simplex* soal 4.3(a).

	Shadow prices	Shadow prices	Shadow prices	Reduced costs	Reduced costs	
y_B	y_1	y_2	y_3	y_4	y_5	c
y_1	1	0	1	0	-1	3
y_4	0	-1	1	1	-1	2
b_1	0	1	1	0	4	(a-12)
	x_3	x_4	x_5	x_1	x_2	
		Slack Variables		Primal Variables		

$$y^* = \begin{bmatrix} y_1^* \\ y_2^* \\ y_3^* \\ y_4^* \\ y_5^* \end{bmatrix} = \begin{bmatrix} 3 \\ 0 \\ 0 \\ 2 \\ 0 \end{bmatrix} \tag{4.72}$$

Untuk mendapatkan solusi optimum untuk variabel x_1 dan x_2 (*primal variables)*, maka nilai tersebut bisa diambil dari baris b pada kolom y_4 dan y_5 yang merupakan bagian dari *opportunity/reduced costs,* yakni,

$$x_1 = 0, x_2 = 4, z = -12 \tag{4.73}$$

b. Perubahan solusi optimum jika terdapat perubahan pada *resources* pada *constraint* pertama menjadi 2,5

Untuk mengakomodasi perubahan ini, maka dapat digunakan hasil *simplex* pada Persamaan (4.72). Perubahan untuk *constraint* pertama dinyatakan pada nilai y_1^* yang bernilai 3 yang diartikan bahwa setiap adanya pengurangan 1 nilai pada *constraint* maka akan menambah hasil *fitness value* sebesar 3 (karena tanda pada *objective function* bertanda minus). Secara matematis, perhitungan tersebut dapat dihitung sebagai berikut.

$$y_1^* = \frac{\Delta z}{\Delta b} = 3 \tag{4.74}$$

$$-x_1 - x_2 \geq -4 \tag{4.75}$$

$$\Delta b = -2{,}5 - (-4{,}0) = 1{,}5$$

$$\Delta z = y_1^* \times \Delta b_1 = 3(1{,}5) = 4{,}5 \quad (4.76)$$

$$z = -12 + 4{,}5 = -7{,}5 \quad (4.77)$$

Dengan demikian, setelah adanya pengurangan resource pada *constraint* pertama menjadi 2,5 akan menyebabkan *fitness value* yang berubah menjadi -7,5 dengan posisi solusi optimum yang tetap. Artinya terjadi penurunan keuntungan dari 12 menjadi 7,5. Meskipun tetap menjadi catatan bahwa penggunaan *shadow price* untuk mengestimasi keuntungan ini hanya dapat dilakukan jika berada dalam rentang yang diizinkan.

c. Perubahan solusi optimum jika terdapat perubahan *resource* pada fungsi kendala ketiga menjadi 1.

Perubahan *resource constraint* ketiga yang semula 5 akan berubah menjadi 1. Untuk kondisi ini, perubahan yang terjadi belum tentu dapat menggunakan nilai y^* dari Persamaan (4.72) karena perubahan pada *shadow price* hanya dapat diaplikasikan untuk perubahan yang bernilai kecil. Untuk memeriksa perubahan ini, dapat dilakukan dengan *Microsoft Excel* seperti yang ditunjukkan pada Bab selanjutnya. Namun, pengecekan perubahan tersebut juga dapat diakomodasi sesuai dengan model dual ini. Matriks A_N^* bisa didapatkan dari Tabel 4.12 untuk solusi non basis, yakni y_2^*, y_3^*, y_5^*.

Tabel 4.12 Matriks akhir metode *simplex* soal 4.3(c).

y_B	y_1	y_2	y_3	y_4	y_5	c
y_1	1	0	1	0	-1	3
y_4	0	-1	1	1	-1	2

$$A_N^* = \begin{bmatrix} a_2^1 & a_3^1 & a_5^1 \\ a_2^4 & a_3^4 & a_5^4 \end{bmatrix} = \begin{bmatrix} 0 & 1 & -1 \\ -1 & 1 & -1 \end{bmatrix} \quad (4.78)$$

Selanjutnya, matriks b_N dan b_B yang telah dimodifikasi bisa didapatkan dari *resource* yang tersedia pada soal sesuai dengan variabel basis dan non basis, dapat ditulis pada persamaan berikut.

$$b_N = [b_2 \quad b_3 \quad b_5] = [1 \quad 1 \quad 0] \quad (4.79)$$

$$b_B = [b_1 \quad b_4] = [4 \quad 0] \quad (4.80)$$

Dengan demikian, matriks b_l dan a^0 sudah dapat dihitung sebagai berikut.

$$b_l = b_N{}' - b_B' A_N^* \tag{4.81}$$

$$b_l = [1 \quad 1 \quad 0] - [4 \quad 0] \begin{bmatrix} 0 & 1 & -1 \\ -1 & 1 & -1 \end{bmatrix} \tag{4.82}$$

$$b_l = [1 \quad -3 \quad 4] \tag{4.83}$$

$$a^0 = [b_1 \quad b_4] \begin{bmatrix} c_1^* \\ c_4^* \end{bmatrix} = [4 \quad 0] \begin{bmatrix} 3 \\ 2 \end{bmatrix} = 12 \tag{4.84}$$

Selanjutnya nilai b_l dan a^0 yang sudah didapatkan dapat dimasukkan kembali ke hasil iterasi terakhir pada metode *simplex* di mana nilai b_l akan mengisi solusi non basis sebelumnya, yakni y_2, y_3, y_5 dan a^0 pada hasil *fitness value.* Perubahan ini dapat dilihat pada tabel berikut.

Tabel 4.13 Perubahan pada matriks *simplex.*

y_B	y_1	y_2	y_3	y_4	y_5	c
y_1	1	0	1	0	-1	3
y_4	0	-1	1	1	-1	2
b_l	0	1	-3	0	4	(a-12)

$b_l = [1 \quad -3 \quad 4]$ $\qquad$ $a^0 = 12$

Dengan metode *simplex*, maka dilakukan iterasi kembali dan dihasilkan tabel sebagai berikut.

Tabel 4.14 Matriks akhir metode *simplex* soal 4.3(c).

y_B	y_1	y_2	y_3	y_4	y_5	c
y_1	1	0	1	0	-1	3
y_2	1	1	0	-1	0	1
b_l	2	0	0	1	1	(a-4)

Berdasarkan tabel tersebut, maka bisa didapatkan hasil solusi optimum yang berubah, yang dapat diambil dari nilai b_l pada kolom y_4 dan y_5.

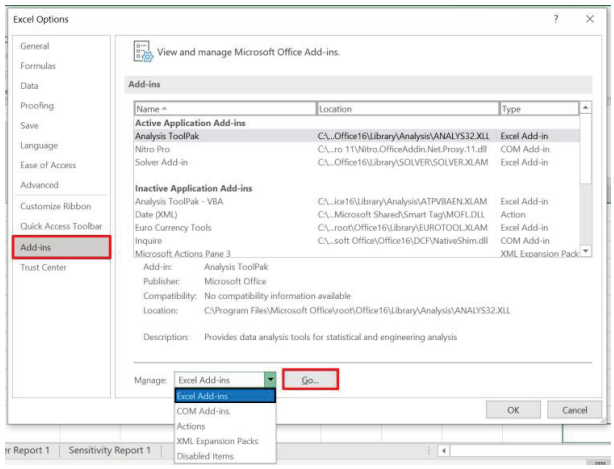
$$x_1 = 1, x_2 = 1, z = -4 \tag{4.85}$$

Dengan demikian, setelah adanya penambahan *resource* pada *constraint* ketiga menjadi 1 akan menyebabkan keuntungan yang berubah menjadi 4 unit biaya dengan posisi solusi optimum $x_1^* = 1, x_2^* = 1$.

4.5. Post Analysis dengan Ms. Excel

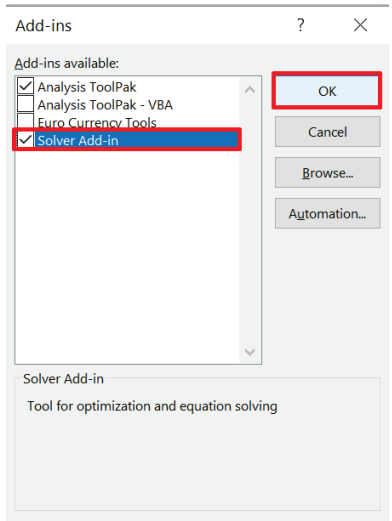
Optimasi dengan metode *simplex* juga dapat dilakukan menggunakan fitur *solver* yang terdapat pada *Microsoft Excel.* Ketersediaan fitur ini dapat dicek pada *tab toolbar,* pilih *data,* kemudian fitur *solver* akan muncul. Jika belum tersedia, maka fitur ini dapat ditambahkan dengan:

1. Klik *File, Option.*
2. Pada jendela baru, pilih *add-ins,* pada bagian *Manage:* pastikan *Excel Add-ins* telah dipilih, kemudian klik *Go.*



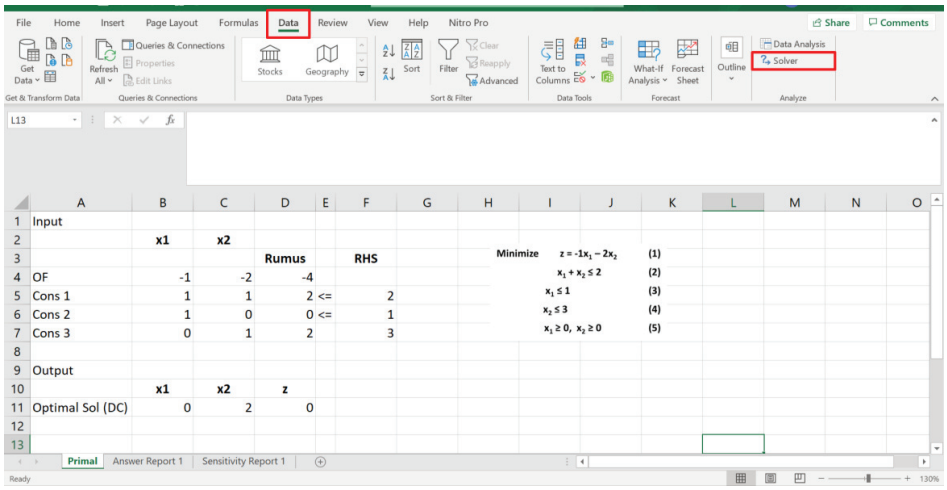
Gambar 4.4 Opsi *add-ins* pada jendela *Excel options.*

3. Pada jendela baru, pastikan *Solver Add-in* sudah di-*check,* kemudian klik OK



Gambar 4.5 Opsi *solver add-in* pada jendela *add-ins.*

4. Fitur *Solver* sudah ditambahkan ke *Microsoft Excel,* periksa kembali fitur *Solver* yang terdapat pada *toolbar Data.*



	A	B	C	D	E	F
1	Input					
2		x1	x2			
3				Rumus		RHS
4	OF	-1	-2	-4		
5	Cons 1	1	1	2	<=	2
6	Cons 2	1	0	0	<=	1
7	Cons 3	0	1	2		3
8						
9	Output					
10		x1	x2	z		
11	Optimal Sol (DC)	0	2	0		
12						
13						

Gambar 4.6 Lokasi fitur *solver* pada *Microsoft Excel.*

Untuk menggunakan fitur *Solver* ini, langkah dan penjelasan akan dijelaskan pada Contoh Soal 4.4 di bawah.

Contoh Soal 4.4

Diketahui suatu permasalahan yang sudah dibuat ke dalam fungsi matematika sebagai berikut.

$$\min z = -1x_1 - 2x_2 \tag{4.86}$$

$$x_1 + x_2 \leq 2 \tag{4.87}$$

$$x_1 \leq 1 \tag{4.88}$$

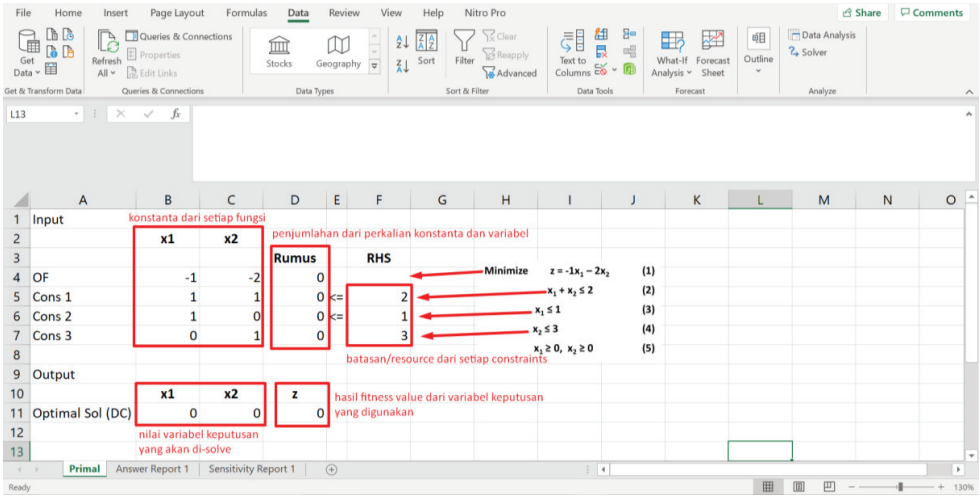
$$x_2 \leq 3 \tag{4.89}$$

$$x_1 \geq 0, x_2 \geq 0 \tag{4.90}$$

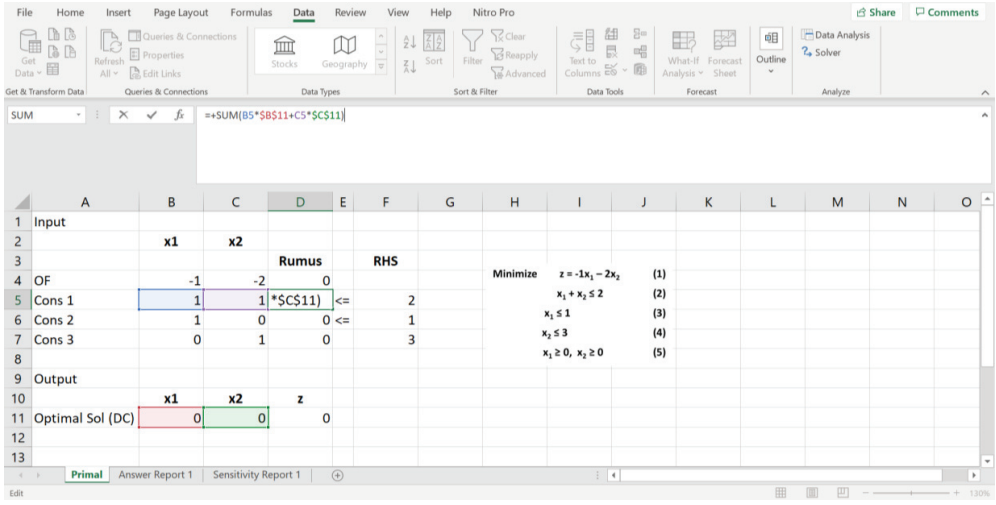
Bagaimana solusi optimum dan *fitness value* dari permasalahan di atas menggunakan fitur *Solver* pada *Microsoft Excel*?

Tahapan pertama yang dapat dilakukan adalah dengan menuliskan permasalahan yang ada pada soal sesuai dengan *format* yang dapat dilihat pada Gambar 4.7. Pada bagian *input,* konstanta untuk setiap fungsi bagi setiap variabel x_1 dan x_2 dituliskan sesuai yang tertera pada soal. Pada bagian kolom RHS, batasan atau *resource* yang tersedia pada soal setelah tanda persamaan atau pertidaksamaan dimasukkan sesuai dengan baris fungsi

masing-masing. Pada bagian *output,* variabel keputusan x_1 dan x_2 dapat diasumsikan nilainya terlebih dahulu, dan kedua nilai ini yang akan diiterasi dan diselesaikan dengan *Solver.* Pada kolom rumus, rumus diisi dengan penjumlahan dari perkalian antara konstanta pada bagian *input* dengan variabel keputusan pada bagian *output.* Contoh penulisan rumus dapat dilihat pada Gambar 4.8 untuk *constraint 1,* sedangkan untuk *constraint* lain dan *objective function* memiliki rumus yang serupa dengan menyesuaikan konstanta untuk setiap barisnya.



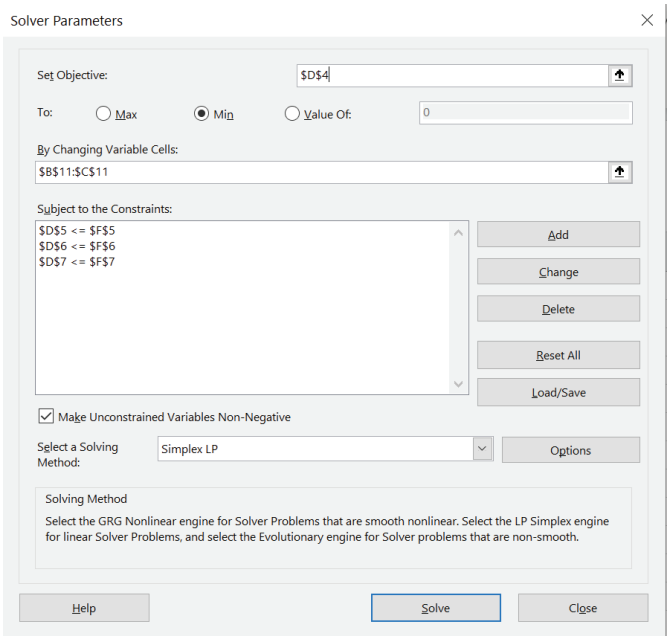
Gambar 4.7 Format penulisan fungsi pada *Microsoft Excel.*



Gambar 4.8 Contoh penjabaran rumus.

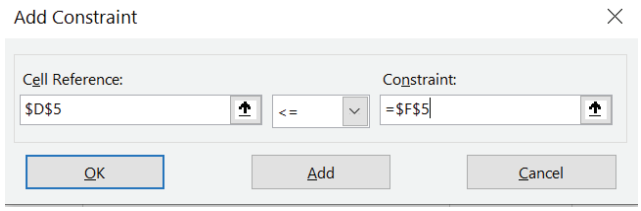
Setelah *format* di atas sudah dituliskan, maka tahap selanjutnya adalah menggunakan fitur *Solver* pada *Toolbar Data.* Setelah muncul jendela baru, maka terdapat

beberapa parameter yang dapat dilihat pada Gambar 4.9 dengan rincian penjelasan di bawah.



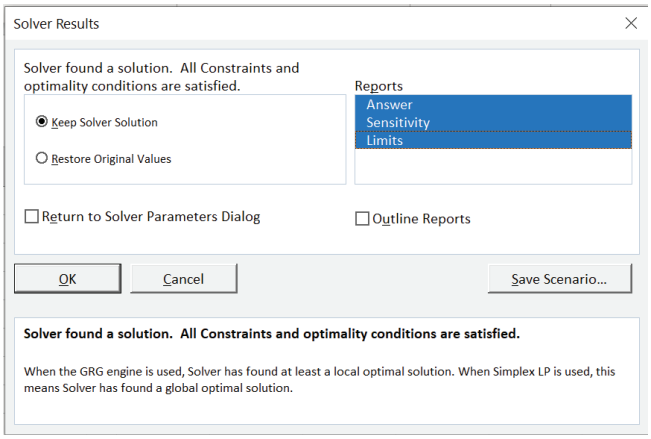
Gambar 4.9 Input solver parameters.

1. *Set Objective:* diisi dengan *cell* rumus pada *objective function.*
2. *To:* dipilih opsi *Min* sesuai dengan fungsi objektif untuk meminimalkan.
3. *By Changing Variable Cells:* diisi dengan variabel keputusan yang akan diiterasi pada bagian *output* sebelumnya.
4. *Subject to the Constraints:* diisi dengan mengklik *Add,* lalu pada bagian *Cell Reference:* diisi dengan *cell* rumus pada bagian *Constraints,* pada bagian tanda disesuaikan dengan tanda yang tertera pada soal, dan pada bagian *Constraint:* disii dengan *cell* pada kolom RHS. Langkah ini dilakukan untuk setiap *constraint* yang terdapat pada soal.



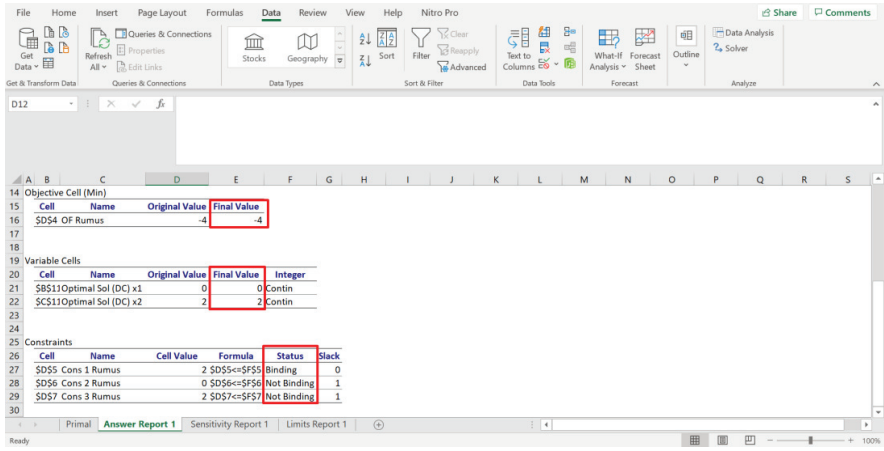
Gambar 4.10 Input Add Constraint.

5. *Select a Solving Method:* pilih *Simplex LP* karena permasalahan ini akan diselesaikan dengan metode *Simplex.*
6. Klik *Solve,* akan muncul jendela baru, pada bagian *Reports* diklik untuk seluruh opsi, yakni *Answer, Sensitivity,* dan *Limits* untuk mendapatkan hasil analisisnya, kemudian klik OK dan proses *solve* akan dilakukan.



Gambar 4.11 Input solver results.

Setelah dilakukan proses di atas, jawaban pada solusi *simplex* di atas terdapat *sheet* baru, *Answer Report 1* yang dapat dilihat pada Gambar 4.12. Berdasarkan hasil tersebut, maka bisa didapatkan hasil *fitness value* yang terdapat pada bagian *Objective Cell (Min)* dengan *final value* sebesar -4. Hasil variabel keputusan didapatkan dari bagian *Variable Cells* dengan *final value* sebesar 0 untuk *cell* x_1 dan 2 untuk *cell* x_2. Selain itu, hasil analisis ini juga menggambarkan *status* dari setiap *constraints,* dan didapatkan bahwa *binding constraint* terdapat pada *constraint* pertama, yang menandakan bahwa solusi optimum berada pada *constraint* tersebut.



Objective Cell (Min)

Cell	Name	Original Value	Final Value
$D$4	OF Rumus	-4	-4

Variable Cells

Cell	Name	Original Value	Final Value	Integer
$B$11	Optimal Sol (DC) x1	0	0	Contin
$C$11	Optimal Sol (DC) x2	2	2	Contin

Constraints

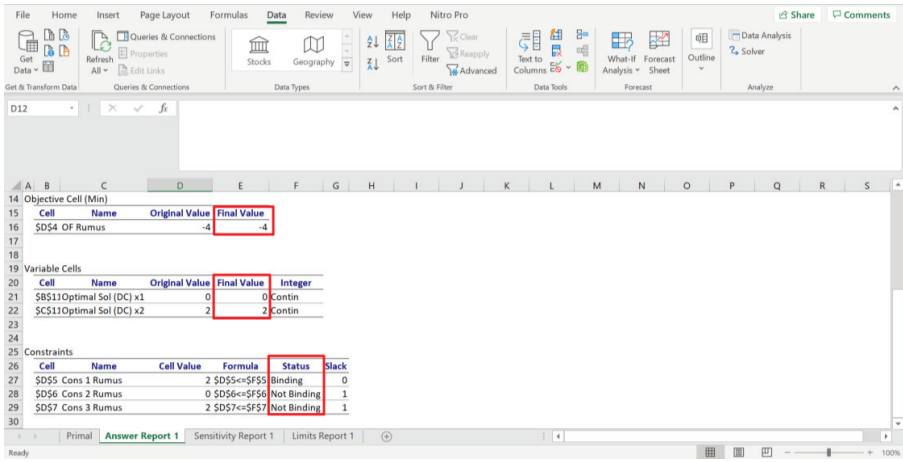
Cell	Name	Cell Value	Formula	Status	Slack
$D$5	Cons 1 Rumus	2	$D$5<=$F$5	Binding	0
$D$6	Cons 2 Rumus	0	$D$6<=$F$6	Not Binding	1
$D$7	Cons 3 Rumus	2	$D$7<=$F$7	Not Binding	1

Gambar 4.12 *Answer report* dari fitur *solver.*

Analisis sensitivitas dapat dilakukan dengan hasil dari solusi *simplex* pada *sheet* baru, *Sensitivity Report 1* yang dapat dilihat pada Gambar 4.13. Hasil analisis sensitivitas yang didapat mencakup *reduced cost* pada bagian *Variable Cells* dan *shadow price* pada bagian *Constraints.* Hasil *reduced cost* didapatkan untuk *non negativity constraint* $x_1 \geq 0$ maupun $x_2 \geq 0$. Nilai *reduced cost* untuk x_1 bernilai 1 yang didapatkan dari kolom *Reduced Cost* yang menandakan bahwa setiap adanya penambahan nilai batas pada *non negativity constraint* akan menyebabkan pengurangan *fitness value* sebesar 1. Ada pun kondisi ini masih dapat diterapkan untuk batas penambahan dan pengurangan yang didapatkan dari kolom *Allowable Increase* dan *Allowable Decrease.* Perubahan pada *non negativity constraint* tersebut dapat diterapkan untuk batas pengurangan nilai sampai dengan 1, sedangkan untuk batas penambahan nilai tidak terbatas. Pada *non negativity constraint* x_2, *reduced cost* bernilai 0 yang menandakan bahwa adanya penambahan atau pengurangan nilai pada *non negativity constraint* tidak akan mengubah nilai *fitness value.* Kondisi ini dapat diterapkan untuk batas penambahan sebesar 1 dan batas pengurangan yang tidak terbatas.

Hasil *shadow price* bisa didapatkan pada bagian *constraints* untuk setiap *constraint* yang ingin dianalisis, yakni dapat diambil dari kolom *Shadow Price.* Sebagai contoh pada *constraint* pertama, nilai *shadow price* adalah -2, yang menandakan bahwa setiap adanya penambahan *resource* pada *constraint* pertama akan mengurangi nilai *fitness value* sebesar 2, dan kondisi ini dapat diterapkan untuk batas penambahan sebesar 1 dan batas pengurangan sebesar 2 yang didapatkan dari kolom *Allowable Increase* dan *Allowable Decrease.* Untuk *constraint* 2 dan 3, besar nilai *shadow price* adalah 0 yang menandakan

bahwa perubahan pada *resource* tidak akan mengubah nilai dari *fitness value* dan terjadi pada batas pengurangan sebesar 1 dan batas penambahan yang tidak terbatas.



Objective Cell (Min)

Cell	Name	Original Value	Final Value
$D$4	OF Rumus	-4	-4

Variable Cells

Cell	Name	Original Value	Final Value	Integer
$B$11	Optimal Sol (DC) x1	0	0	Contin
$C$11	Optimal Sol (DC) x2	2	2	Contin

Constraints

Cell	Name	Cell Value	Formula	Status	Slack
$D$5	Cons 1 Rumus	2	$D$5<=$F$5	Binding	0
$D$6	Cons 2 Rumus	0	$D$6<=$F$6	Not Binding	1
$D$7	Cons 3 Rumus	2	$D$7<=$F$7	Not Binding	1

Gambar 4.13 Sensitivity report dari fitur solver.

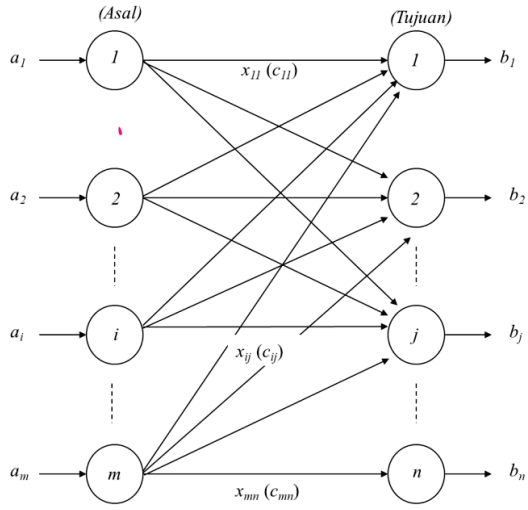
Pembaca yang tertarik untuk melihat penjelasan dalam bentuk audio-visual terkait Bab ini dapat mengunjungi playlist youtube berikut:

https://bit.ly/Playlist-LPPostAnalysis

5. *Transportation Problem*

5.1. Pengantar

Transportation problem merupakan salah satu varian dari *linear programming*, yang pada dasarnya bertujuan untuk mendistribusikan barang dari titik asal menuju titik tujuan. Permasalahan yang ada pada *transportation problem* dimodelkan dalam suatu jaringan yang terdiri dari lokasi sumber (m) dan lokasi tujuan (n) yang direpresentasikan dalam titik yang kemudian dihubungkan oleh *link* yang berisi dua informasi, yakni unit biaya transportasi (i.e., e_{ij}) dan jumlah dari barang yang dikirimkan (i.e., x_{ij}). Selain itu, permasalahan ini juga dibatasi oleh kendala batasan dari sisi jumlah barang yang tersedia pada penyedia-*i* (i.e., a_i) pada lokasi asal dan jumlah barang yang dibutuhkan pada tujuan-*j* (i.e. b_j). Tujuan dari model ini adalah untuk menentukan jumlah yang akan ditransportasikan dari masing-masing lokasi sumber ke lokasi tujuan dengan total biaya minimal. Selain itu, permasalahan ini juga tetap memperhatikan adanya fungsi kendala dari sisi jumlah barang yang tersedia di lokasi sumber dan barang yang dibutuhkan pada lokasi tujuan. Ilustrasi dari model dari *transportation problem* dapat dilihat pada Gambar 5.1.



(dimodifikasi dari Taha, 2007).

Gambar 5.1 Ilustrasi Transportation Problem.

5.2. *Balanced Transportation Problem*

Dalam bentuk fungsi matematika, ilustrasi permasalahan tersebut dapat dideskripsikan sebagai berikut.

$$\min z = \sum\sum c_{ij}x_{ij} \tag{5.1}$$

$$\sum_{j=1}^{n} x_{ij} \leq a_i, \forall\, i = 1,2,\dots,m \tag{5.2}$$

$$\sum_{i=1}^{m} x_{ij} \geq b_j, \forall\, j = 1,2,\dots,n \tag{5.3}$$

$$x_{ij} \geq 0 \tag{5.4}$$

Berdasarkan Persamaan (5.2) dan (5.3) dapat dilihat bahwa fungsi kendala berusaha memastikan bahwa jumlah yang tersedia pada lokasi asal harus lebih besar atau sama dengan barang yang akan dikirimkan dari lokasi asal (Persamaan (5.2)). Sementara Persamaan (5.3) memastikan bahwa jumlah yang dikirimkan (i.e., total x_{ij}) kepada lokasi tujuan harus lebih besar dibandingkan jumlah yang dibutuhkan. Persamaan (5.4) memastikan bahwa jumlah barang yang dikirimkan bersifat *non-negative*.

Namun, agar model yang harus dipecahkan menjadi lebih sederhana, maka kendala pertidaksamaan ditransformasikan menjadi bentuk persamaan, dengan memastikan bahwa:

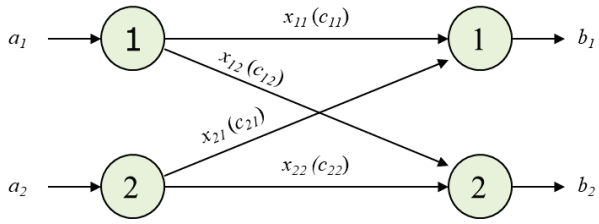
- Jumlah yang dikirimkan dari lokasi sumber sama dengan yang diterima oleh lokasi tujuan (i.e., $\sum a_i = \sum b_j$).
- Jumlah barang yang dikirimkan dari lokasi asal ke lokasi tujuan harus sama dengan total persediaan dari lokasi asal (lihat Persamaan (5.5))
- Jumlah barang yang dikirimkan ke lokasi tujuan dari lokasi asal harus sama dengan total kebutuhan yang dibutuhkan di lokasi tujuan (lihat Persamaan (5.6))

Adanya asumsi ini membuat permasalahan *transportation* menjadi permasalahan yang seimbang dari sisi persediaan dan kebutuhan atau umum disebut *balanced transportation model*.

$$\sum_{j=1}^{n} x_{ij} = a_i, \forall\, i = 1,2,\dots,m \tag{5.5}$$

$$\sum_{i=1}^{m} x_{ij} = b_j, \forall\, j = 1,2,\dots,n \tag{5.6}$$

Sebagai ilustrasi sederhana dari permasalahan ini, diberikan contoh untuk permasalahan transportasi yang melibatkan dua lokasi sumber/asal dan dua lokasi tujuan seperti terlihat pada Gambar 5.2.



Gambar 5.2 Ilustrasi *Balanced Transportation Problem* (Taha, 2007).

Berdasarkan ilustrasi di atas, maka dapat dijelaskan bahwa terdapat dua lokasi sumber a_1 dan a_2 serta dua lokasi tujuan b_1 dan b_2 yang dihubungkan oleh setiap *link.* Pada *balanced transportation model,* maka jumlah persediaan dari a_1 dan a_2 harus memiliki jumlah yang sama dari jumlah kebutuhan yang ada di b_1 dan b_2. Tujuan utama dari model ini adalah untuk menentukan jumlah barang yang didistribusikan dari setiap pasangan lokasi sumber dan lokasi tujuan yang direpresentasikan oleh *link* $(x_{11}, x_{12}, x_{21}, x_{22})$ dengan biaya total serendah mungkin. Permasalahan di atas dapat ditulis menjadi persamaan berikut.

$$\min z = c_{11}x_{11} + c_{12}x_{12} + c_{21}x_{21} + c_{22}x_{22} \tag{5.7}$$

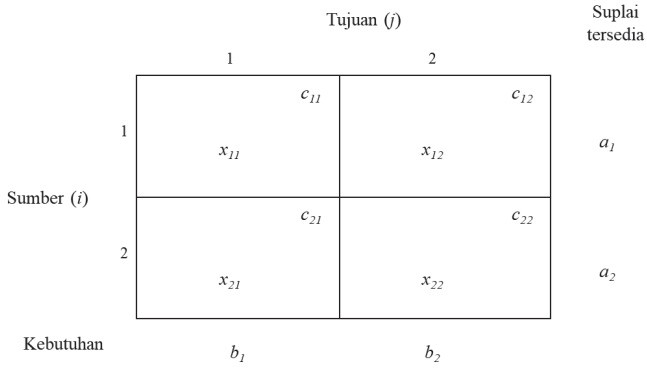
$$x_{11} + x_{12} = a_1 \tag{5.8}$$

$$x_{21} + x_{22} = a_2 \tag{5.9}$$

$$x_{11} + x_{21} = b_1 \tag{5.10}$$

$$x_{12} + x_{22} = b_2 \tag{5.11}$$

$$x_{11}, x_{12}, x_{21}, x_{22} \geq 0 \tag{5.12}$$



Gambar 5.3 Format penyelesaian *transportation problem* (Taha, 2007).

Untuk menyelesaikan permasalahan ini, dapat digunakan metode *simplex*. Meskipun untuk memudahkan proses perhitungan permasalahan transportasi ini dapat diselesaikan dengan menggunakan tabel seperti pada Gambar 5.3. Prosedur penggunaan tabel ini dapat dijabarkan sebagai berikut.

1. Tentukan *initial feasible solution*, yang dapat dilakukan dengan mengaplikasikan metode *northwest, least cost,* atau *vogel approximation method* (VAM).
2. Gunakan solusi awal dari tahap 1 untuk menentukan variabel yang akan dimasukkan sebagai variabel basis dan non basis. Evaluasi apakah kondisi optimum sudah tercapai, jika belum maka lanjutkan ke tahap 3.
3. Tentukan variabel basis yang akan dikeluarkan serta variabel non basis yang akan dimasukkan sebagai solusi. Lanjutkan ke tahap 2 dan proses tersebut diulang hingga kondisi optimum tercapai.

Untuk memberikan ilustrasi yang lebih baik, prosedur perhitungan dapat dilihat pada **Contoh Soal 5.1.**

Contoh Soal 5.1

Suatu perusahaan *ready mix concrete* akan mengirimkan *ready mix concrete* ke 4 lokasi konstruksi yang berbeda. Setiap kebutuhan dari lokasi konstruksi tersebut dapat dilihat pada Tabel 5.1. Perusahaan memiliki 3 *batching plants,* di mana setiap *batching plants* memiliki kapasitas dan biaya transportasi yang berbeda, dapat dilihat pada Tabel 5.2. Biaya dari setiap *batching plant* menuju lokasi konstruksi dapat dilihat pada Tabel 5.3. Tentukan jumlah produk yang perlu dikirimkan dari setiap *batching plant* ke lokasi konstruksi yang paling optimal menggunakan metode *northwest, least cost,* dan VAM.

Tabel 5.1 Kebutuhan setiap lokasi konstruksi.

Lokasi Konstruksi	Kebutuhan
1	10
2	20
3	20
4	30

Tabel 5.2 Kapasitas produksi dari *Batching Plant.*

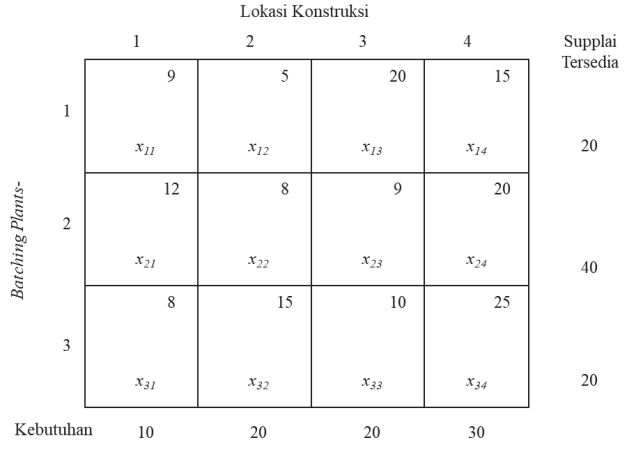
Batching Plant	Kapasitas Produksi
1	20
2	40
3	20

Tabel 5.3 Kapasitas produksi dari setiap *Batching Plant.*

Pasangan Titik (*Batching plant*-Lokasi konstruksi)	Biaya Transportasi
1-1	10
1-2	5
1-3	20
1-4	15
2-1	12
2-2	8
2-3	9
2-4	20
3-1	8
3-2	15
3-3	10
3-4	25

Tahap 1 Solusi Awal

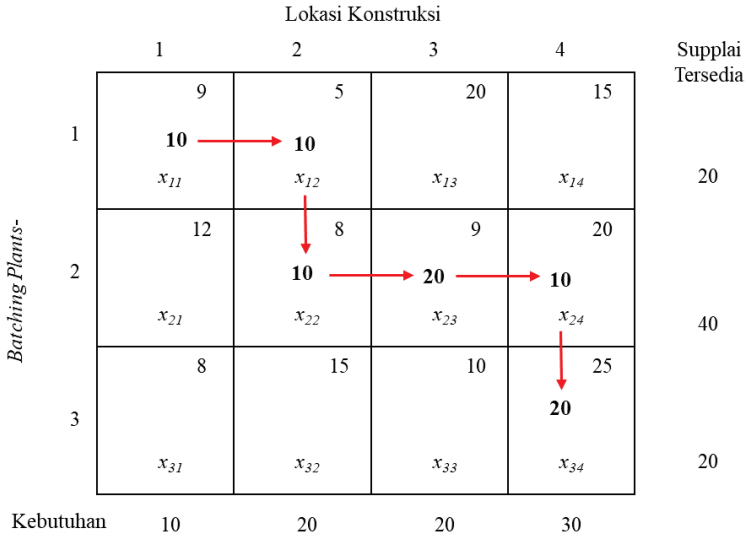
Permasalahan di atas dapat disusun menjadi Gambar 5.4 di mana setiap kolom diisi dengan jumlah barang yang dikirimkan dari sumber ke lokasi tujuan (x_{ij}) dengan biaya yang telah ditentukan (c_{ij}). Total persediaan yang dimiliki masing-masing *batching plant* berada pada kolom paling akhir, sementara total kebutuhan yang dibutuhkan masing-masing lokasi berada di baris paling akhir dari tabel tersebut. Solusi awal dapat ditentukan dengan menggunakan salah satu metode berikut *north west, least cost,* dan VAM, meskipun pada buku ini setiap metode akan dicoba untuk memberikan gambaran penggunaan ketiga metode tersebut.



Gambar 5.4 Penyusunan permasalahan.

5.2.1. Solusi awal dengan metode *North West*

Pada metode *northwest,* tahap pertama yang dilakukan adalah dengan mengalokasikan *resource* sebanyak mungkin pada sudut barat laut tabel (*north west corner*). Pada *north west corner,* maka jumlah yang dibutuhkan oleh lokasi konstruksi 1 adalah sebanyak 10, sehingga pada kotak x_{11} dialokasikan sebanyak 10. Karena lokasi konstruksi-1 telah terpenuhi, proses distribusi dilanjutkan pada lokasi konstruksi-2. Jika dilihat dari *batching plants* 1, total *supply* yang akan disediakan adalah sejumlah 20. Karena sudah terisi 10 pada lokasi konstruksi 1, maka jumlah yang dialokasikan pada lokasi konstruksi*s 2* adalah sejumlah 10.



Gambar 5.5 *Initial solution* pada metode *North West.*

Selanjutnya dapat dilihat dari lokasi konstruksi 2 memiliki kekurangan sebesar 10, sehingga sisanya akan dipenuhi dari *batching plants* 2 sejumlah 10. Sehingga kebutuhan pada lokasi konstruksi-2 telah terpenuhi sejumlah 20. Tahapan selanjutnya adalah berusaha memenuhi kebutuhan lokasi konstruksi-3 dengan kebutuhan 20. Pada *batching plants* 2, total *supply* yang disediakan adalah sejumlah 40, karena sudah terisi 10 untuk lokasi konstruksi 2, maka sisa yang dapat dialokasikan untuk lokasi konstruksi 3 adalah sebanyak 20. Selanjutnya, untuk lokasi konstruksi 4, total *demand* yang dibutuhkan adalah sejumlah 30. Bagian pertama dapat dialokasikan dari *plant-2* sebanyak 10 sisanya dialokasikan dari *batching plants* 3 sejumlah 20. Perhitungan di atas diilustrasikan pada Gambar 5.5. Berdasarkan gambar di atas, maka didapatkan solusi basis untuk solusi awal dari permasalahan ini, yakni sebagai berikut.

$$x_{11} = 10, x_{12} = 10, x_{22} = 10, x_{23} = 20, x_{24} = 10, x_{34} = 20 \quad (5.13)$$

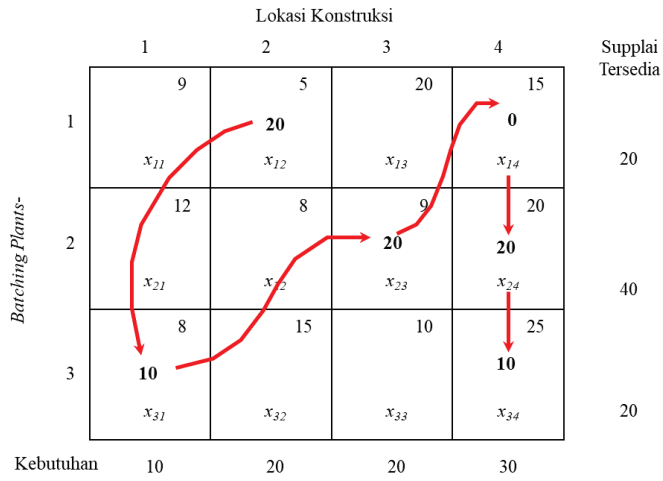
Nilai *fitness* dari solusi awal ini adalah sebagai berikut.

$$z = (9 \times 10) + (5 \times 10) + (8 \times +10) + (9 \times 20) + (20 \times 10) + (25 \times 20) \quad (5.14)$$
$$= 1{,}100$$

5.2.2. Solusi awal dengan metode *Least Cost*

Pada metode *least cost,* pemilihan solusi dimulai dari kotak yang memiliki biaya yang paling murah. Pada permasalahan di atas, maka solusi pertama dimulai dari biaya yang paling murah, yakni pada kotak x_{12}. Pada kotak tersebut, dapat dilihat bahwa persediaan dari *batching plants* 1 adalah sebesar 20 sementara kebutuhan untuk *construction* 2 juga bernilai 20, dengan demikian, dialokasikan sebanyak-banyaknya ke kotak tersebut (i.e., 20).

Selanjutnya, cari kembali kotak dengan biaya yang terkecil, yakni pada kotak x_{31} dengan biaya transportasi sebesar 8. Kotak x_{22} memiliki besar biaya yang sama, akan tetapi lokasi konstruksi-2 telah terpenuhi sehingga dipilih kotak x_{31}. Pada kotak tersebut, dapat dilihat bahwa *batching plant* 3 dapat menyediakan 20, namun lokasi konstruksi 1 hanya memiliki kebutuhan sejumlah 10, sehingga dialokasikan 10 pada kotak tersebut. Tahap selanjutnya adalah dengan mencari kembali biaya yang termurah ketiga, yakni pada kotak x_{23}. Kebutuhan pada lokasi konstruksi tersebut adalah 20 sementara *batching plant*-2 menyediakan 40, sehingga 20 akan dialokasikan kepada kotak x_{23}.



Gambar 5.6 *Initial solution* pada metode *Least Cost.*

Tahapan selanjutnya adalah mencari alokasi untuk lokasi konstruksi-4, dimana kotak termurah berada pada x_{14}. Namun, pada kotak tersebut, seluruh alokasi yang tersedia telah digunakan menuju lokasi konstruksi 2. Sehingga, dicari kembali biaya yang termurah setelahnya, yakni kotak x_{24} dengan biaya 20. Pada kotak ini, terdapat sisa persediaan dari *batching plant* 2 sebanyak 20 dan kebutuhan pada lokasi konstruksi *4* sebanyak 30, maka dialokasikan 20 pada kotak ini. Nilai biaya terkecil selanjutnya adalah pada kotak x_{34} dengan biaya 25, persediaan yang tersisa pada *plant-3* kemudian dialokasikan seluruhnya kepada lokasi-4. Perhitungan di atas dapat dilihat pada Gambar 5.6.

Berdasarkan tabel diatas dapat diketahui bahwa, solusi awal dan nilai *fitness* dapat dihitung seperti pada Persamaan (5.15) dan (5.16). Dapat terlihat bahwa metode *least cost* memberikan biaya yang lebih rendah jika dibandingkan *northwest method*, karena metode *least cost* berusaha untuk mengalokasi persediaan ke kotak dengan biaya terendah terlebih dahulu.

$$x_{12} = 20, x_{31} = 10, x_{23} = 20, x_{14} = 0, x_{34} = 10, x_{24} = 20 \tag{5.15}$$

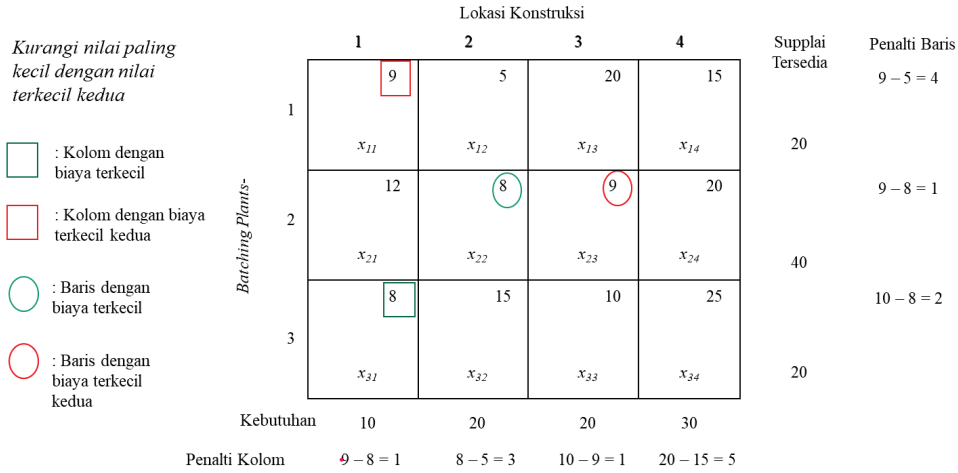
$$\begin{aligned} z &= (20 \times 5) + (10 \times 8) + (20 \times 9) + (0 \times 15) + (20 \times 20) + (10 \times 25) \\ &= 1090 \end{aligned} \tag{5.16}$$

5.2.3. Solusi awal dengan metode *Vogel Approximation Method* (VAM)

Metode lain yang dapat digunakan untuk memperoleh solusi awal adalah dengan menggunakan VAM. Prinsip dari metode ini adalah mempertimbangkan tidak hanya biaya terkecil, akan tetapi pula bagaimana jika tidak bisa mengalokasikan barang pada kotak dengan biaya terkecil, yang direpresentasikan dengan *penalty* baris atau kolom. Penalti ini memberikan ilustrasi terkait kenaikan biaya yang akan dirasakan ketika tidak dialokasikannya barang pada kotak dengan biaya terkecil. Nilai penalti dihitung dengan mengurangkan nilai biaya terkecil dengan biaya terkecil kedua pada baris atau kolom yang sama. Selanjutnya, baris atau kolom yang memiliki nilai penalti terbesar dipilih, dan pada baris atau kolom terpilih tersebut diberikan alokasi sebanyak-banyaknya kepada kotak yang memiliki biaya terkecil. Adapun tahapan dalam operasionalisasinya adalah sebagai berikut.

a. Mencari nilai penalti pada setiap baris dan kolom

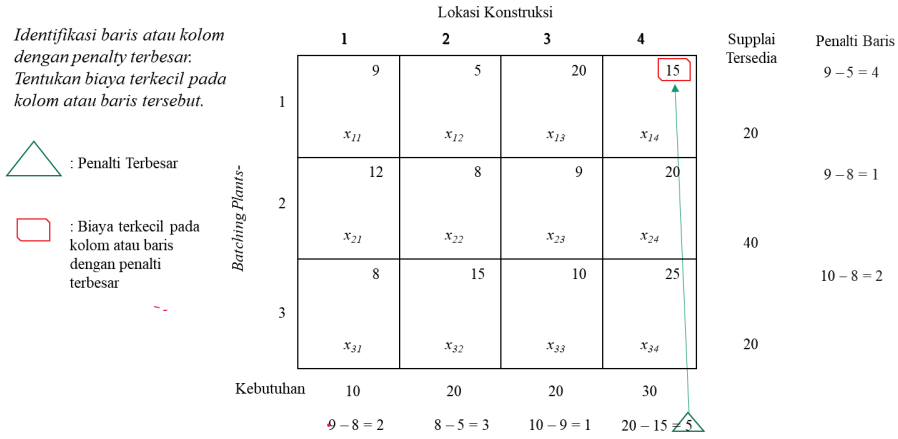
Setiap kolom dan baris akan dihitung penaltinya dengan mengurangi nilai terkecil kedua dengan nilai yang paling kecil.



Gambar 5.7 Identifikasi nilai penalti pada setiap baris dan kolom.

b. Mengidentifikasikan baris atau kolom dengan nilai penalti terbesar.

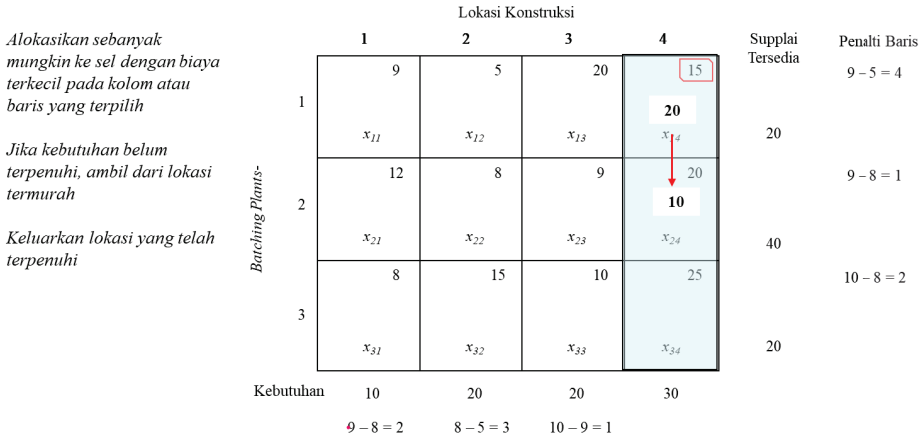
Nilai penalti terbesar dari baris dan kolom di atas adalah 5 pada kolom 4 (Lihat Gambar 5.8). Pada kolom 4, biaya termurah ada pada kotak x_{14} dengan biaya 15. Dengan demikian, pengisian kotak dimulai dari kotak tersebut, yakni sejumlah 20 untuk memenuhi kebutuhan lokasi konstruksi 4. Karena kebutuhan lokasi-4 adalah 30, maka pemenuhan kebutuhan diambil dari lokasi yang memiliki biaya paling rendah, yaitu kotak x_{24}. Dengan konfigurasi ini, maka lokasi-4 telah terpenuhi kebutuhannya, serta dapat dikeluarkan dari proses perhitungan (lihat Gambar 5.9).



Gambar 5.8 Identifikasi baris atau kolom dengan penalti terbesar.

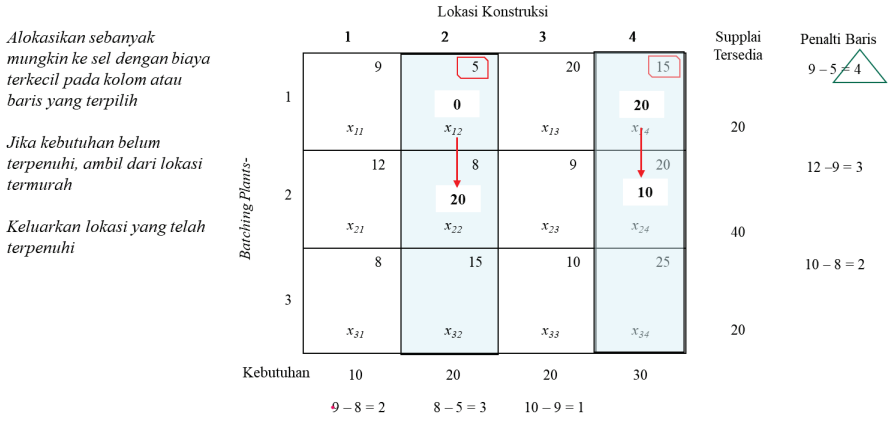
Selanjutnya dilakukan kembali perhitungan penalti untuk setiap baris dan kolom. Pada tahapan ini, nilai penalti adalah sebesar 4 dari baris 1. Pada baris tersebut, biaya termurah terdapat pada kotak x_{12} dengan biaya 5. Meskipun karena *plant*-1 telah habis

persediaannya, maka diambil dari *plant* dengan biaya transportasi termurah, yaitu pada kotak x_{12}. Pada kotak tersebut, dialokasikan sejumlah 20 untuk memenuhi kebutuhan lokasi konstruksi 2 sebesar 20 dan memenuhi persediaan *batching plant* 1 sebesar 20 (Lihat Gambar 5.10).



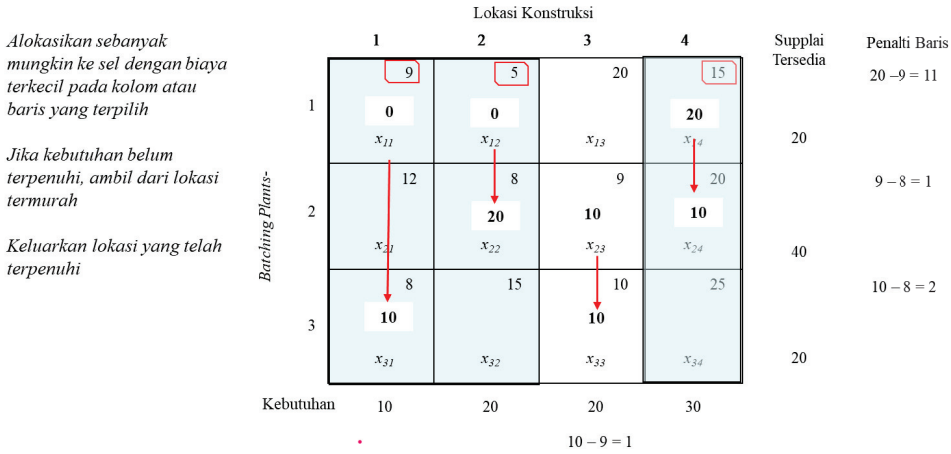
Gambar 5.9 Identifikasi penalti terbesar setelah kolom lokasi-4 ditutup.

Karena kebutuhan pada lokasi konstruksi 2 sudah terpenuhi, maka kolom tersebut juga dapat ditutup dan selanjutnya dilakukan kembali perhitungan penalti untuk setiap baris dan kolom yang tersisa. Pada tahapan ini, nilai penalti adalah sebesar 11 dari baris 1 (lihat Gambar 5.11). Pada baris tersebut, biaya termurah terdapat pada kotak x_{23} dengan biaya 9. Karena *batching plant* -1 telah habis persediaannya, maka dialokasi ke *plant* termurah lainnya, yaitu *plant* 3. Pada kotak kotak x_{31} tersebut, dialokasikan sejumlah 10 untuk memenuhi kebutuhan lokasi konstruksi 3.



Gambar 5.10 Identifikasi penalti terbesar setelah kolom lokasi-2 ditutup.

Karena kebutuhan pada lokasi konstruksi 1 sudah terpenuhi, maka kolom tersebut juga dapat ditutup dan hanya tersisa kolom 3 (lihat Gambar 5.11). Kolom terakhir yang tersisa adalah kolom untuk lokasi konstruksi-3. Pada kolom ini, biaya termurah terdapat pada kotak x_{23} dengan biaya 9. Namun pada kotak tersebut, ketersediaan dari *batching plant 2* hanya tersisa 10, sehingga sisanya harus diambil dari *batching plant 3*.



Gambar 5.11 Identifikasi *penalty* terbesar setelah kolom ketiga ditutup.

Berdasarkan gambar di atas, maka bisa didapatkan solusi awal dengan menggunakan metode VAM di atas, dapat dilihat pada persamaan (5.17) dan Persamaan (5.18). Dari persamaan tersebut terlihat bahwa solusi yang diberikan oleh VAM lebih rendah dibandingkan kedua solusi sebelumnya. Karena karakteristiknya yang dapat memberikan solusi yang cukup baik, VAM pada banyak kasus, terutama dengan skala kasus yang besar, umum digunakan sebagai alat hitung utama. Kondisi disebabkan karena penggunaan metode eksak sulit untuk diterapkan, sehingga VAM menjadi pilihan yang rasional.

$$x_{14} = 20, x_{22} = 20, x_{23} = 10, x_{24} = 10, x_{31} = 10, x_{33} = 10 \tag{5.17}$$

Fitness value dari *initial basic solution* di atas dapat dihitung sebagai berikut.

$$\begin{aligned} z &= (20 \times 15) + (20 \times 8) + (10 \times 9) + (10 \times 20) + (10 \times 8) + (10 \times 10) \\ &= 930 \end{aligned} \tag{5.18}$$

5.2.4. Penentuan Variabel Basis dan Non-Basis

Tahap ini digunakan untuk menentukan variabel yang akan keluar dan masuk untuk mendapatkan solusi optimum dari permasalahan. Jumlah solusi basis bisa didapatkan dari

persamaan $m + n - 1$, di mana m merupakan jumlah asal, n merupakan jumlah tujuan. Pada kasus soal ini, maka jumlah solusi basis adalah berjumlah 6 (i. e. , $3 + 4 - 1$). Untuk menentukan variabel yang masuk dan keluar, maka digunakan *method of multiplier.* Untuk setiap variabel x_{ij}, biaya dari kotak tersebut dapat dihitung menggunakan Persamaan (5.19).

$$U_i + V_i = c_{ij} \tag{5.19}$$

Pada penyelesaian soal ini, diasumsikan penyelesaian solusi ini akan menggunakan solusi awal yang sudah dihitung menggunakan *North West Corner Method.* Berdasarkan Persamaan (5.19), maka didapatkan bentuk persamaan untuk setiap variabel basis yang dapat dilihat pada kolom 2 pada Tabel 5.4. Nilai c_{ij} didapatkan berdasarkan besar biaya pada kotak solusi basis yang ditinjau. Untuk solusi dari setiap nilai U_i dan V_j, diasumsikan terlebih dahulu nilai $U_i = 0$, dan nilai U_i dan V_j pada solusi lainnya juga dapat ditentukan. Sebagai contoh pada variabel basis x_{11}, diasumsikan $U_i = 0$, maka $V_1 = 9$. Dengan demikian, nilai U_i dan V_j untuk setiap baris dan kolom dapat diketahui.

Tabel 5.4 Penentuan nilai *U* dan *V* tahap 1.

Variabel basis	(*U*,*V*)	Solusi
x_{11}	$U_1 + V_1 = 9$	Set U_1=0 → V_1=9
x_{12}	$U_1 + V_2 = 5$	U_1= 0 → V_2= 5
x_{22}	$U_2 + V_2 = 8$	V_2= 5→ U_2= 3
x_{23}	$U_2 + V_3 = 9$	U_2= 3 → V_3= 6
x_{24}	$U_2 + V_4 = 20$	U_2= 3 → V_4= 17
x_{34}	$U_3 + V_4 = 25$	V_4= 17 → U_3= 8

Tahap selanjutnya adalah dengan mengevaluasi variabel non basis dengan melihat nilai $U_i + V_j - c_{ij}$ untuk setiap variabel non basis. Nilai U_i dan V_j didapatkan dari Tabel 5.4 yang sudah dihitung sebelumnya, dan c_{ij} untuk setiap variabel non basis ditentukan berdasarkan biaya pada kotak yang ditinjau. Evaluasi dilakukan untuk setiap variabel non basis dan dapat dilihat pada Tabel 5.5. Selanjutnya, nilai hasil evaluasi tersebut akan direkap pada tabel z yang dapat dilihat pada Tabel 5.6, di mana nilai hasil evaluasi dimasukkan untuk setiap variabel non basis, dan untuk variabel basis diberikan nilai 0.

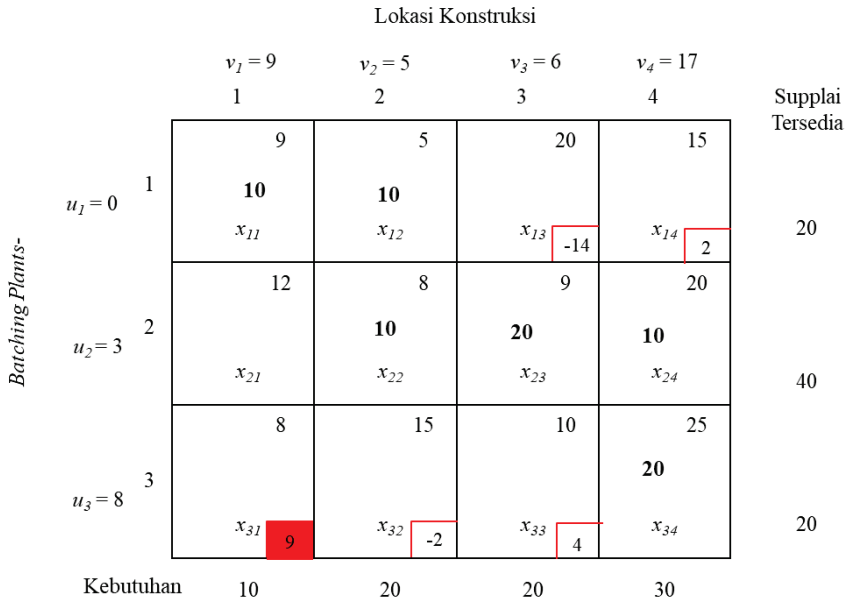
Tabel 5.5 Evaluasi variabel non basis tahap 1.

Variabel non basis	$U_i+V_j- c_{ij}$
x_{13}	$U_1+V_3-C_{13}$=0+6-20= -14
x_{14}	$U_1+V_4-C_{14}$= 0+17 -15= 2
x_{21}	$U_2+V_1-C_{21}$= 3+9-12= 0
x_{31}	$U_3+V_1-C_{31}$=8+9-8= 9
x_{32}	$U_3+V_2-C_{32}$=8+5-15= -2
x_{33}	$U_3+V_3-C_{33}$=8+6-10= 4

Tabel 5.6 Rekapitulasi nilai *z* setiap variabel tahap 1.

Basic	x_{11}	x_{12}	x_{13}	x_{14}	x_{21}	x_{22}	x_{23}	x_{24}	x_{31}	x_{32}	x_{33}	x_{34}
Z	0	0	-14	2	0	0	0	0	9	-2	4	0

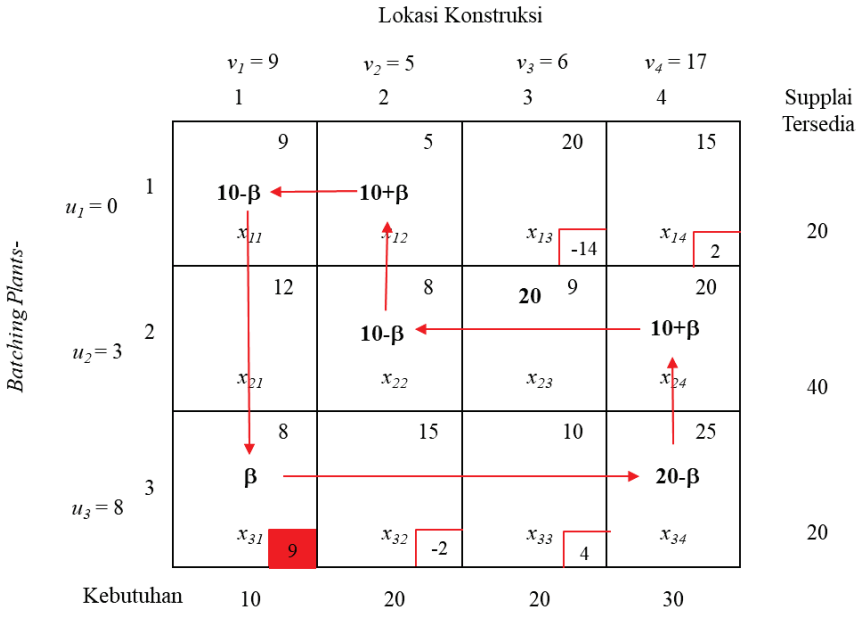
Berdasarkan hasil evaluasi pada tabel *z*, maka variabel non basis yang akan masuk ke variabel basis adalah variabel yang memiliki nilai positif terbesar, yakni variabel x_{31} dengan nilai 9. Nilai ini menyatakan setiap ditambahkannya nilai kuantitas pada variabel x_{31}, maka akan mengurangi total biaya sebesar 9. Untuk mempermudah analisis selanjutnya, maka nilai *z* pada setiap variabel non basis ini juga ditambahkan kepada matriks asal tujuan yang sudah berisi solusi basis sebelumnya, dapat dilihat pada Gambar 5.12.



Gambar 5.12 Matriks asal tujuan beserta nilai *z*.

Untuk menentukan variabel basis yang keluar, maka akan dilakukan *reduction process* dengan membentuk sebagai *loop* yang dimulai dan diakhiri dari variabel non basis yang akan masuk, yakni x_{31}. Proses *loop* hanya mengandung koneksi berupa horizontal dan vertikal saja, tanpa garis diagonal, dengan aturan tanda $+ \rightarrow\ - \rightarrow + \rightarrow\ - \rightarrow +$. Proses ini dapat dilihat pada Gambar 5.13.

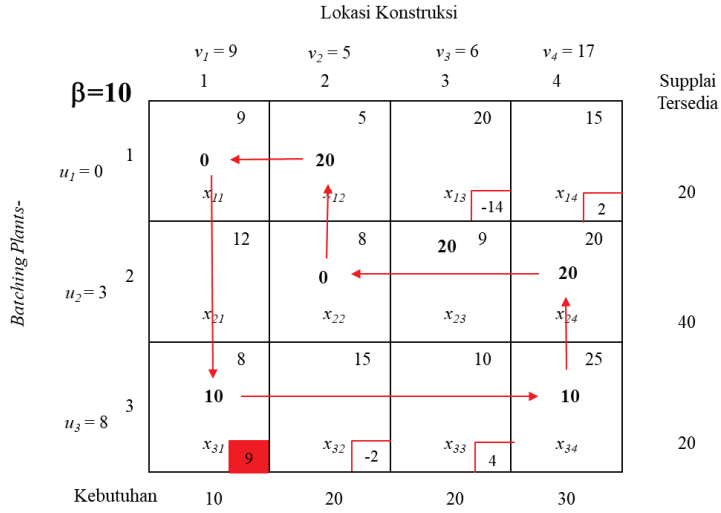
Tahapan ini dimulai dengan mengasumsikan nilai pada variabel non basis yang akan masuk sejumlah β. Karena sudah ditambahkan nilai β, maka nilai kotak pada x_{34} yang semula 20 akan berubah menjadi $20 - \beta$. Selanjutnya karena sudah terisi $20 - \beta$, maka nilai kotak pada x_{24} yang semula 10 juga berubah menjadi $10 + \beta$ sesuai pergerakan *loop* ke atas berupa vertikal dan tanda yang berubah menjadi positif. Proses *loop* selanjutnya dapat dilakukan secara horizontal menuju x_{23}. Namun, jika diambil variabel ini, *loop* akan berhenti karena tidak bisa diteruskan secara vertikal menuju variabel basis lainnya. Dengan demikian, diambil variabel basis selanjutnya pada x_{22} yang akan berubah menjadi $10 - \beta$. Proses selanjutnya diteruskan secara vertikal menuju variabel x_{12} menjadi $10 + \beta$, horizontal menuju variabel x_{11} menjadi $10 - \beta$, dan vertikal kembali ke variabel x_{31} dan *loop* ditutup.



Gambar 5.13 *Reduction process* dengan nilai $\boldsymbol{\beta}$ tahap 1.

Untuk menentukan nilai β, maka ditentukan nilai maksimum yang dapat ditentukan pada kotak yang berisi nilai β, yakni pada variabel x_{31} agar mampu mengurangi total biaya

sebesar-besarnya. Nilai yang dapat diisi pada variabel ini adalah sejumlah 10 sesuai dengan kebutuhan pada kolom 1. Dengan demikian, jumlah pada setiap kotak hasil pertambahan dan pengurangan dengan β dapat dilihat pada Gambar 5.14.



Gambar 5.14 Nilai matriks dengan $\beta = 10$ tahap 1.

4.2.5. Evaluasi Kondisi Optimalitas

Tahapan selanjutnya adalah dengan mengevaluasi apakah masih terdapat nilai *z* yang positif pada variabel non basis pada hasil terakhir yang didapat. Jika masih terdapat nilai *z* yang positif maka langkah dilakukan pada tahapan sebelumnya dilakukan kembali (i.e., penentuan variabel basis dan non basis). Langkah evaluasi sama seperti pada tahapan sebelumnya, yakni dengan menentukan nilai U_i dan V_j dari variabel basis yang ada, dapat dilihat pada Tabel 5.7, kemudian dilakukan evaluasi *z* pada variabel non basis pada Tabel 5.8 dan nilai *z* pada setiap variabel direkap pada Tabel 5.9.

Tabel 5.7 Penentuan nilai U dan V tahap 2.

Variabel Basis	(*U*,*V*)	Solusi
x_{12}	$U_1 + V_2 = 5$	Set U_1=0 → V_2=5
x_{22}	$U_2 + V_2 = 8$	V_2= 5 → U_2= 3
x_{23}	$U_2 + V_3 = 9$	U_2= 3 → V_3= 6
x_{24}	$U_2 + V_4 = 20$	U_2= 3 → V_4= 17
x_{34}	$U_3 + V_4 = 25$	V_4= 17 → U_3= 8
x_{31}	$U_3 + V_1 = 8$	U_3= 8 → V_1=0

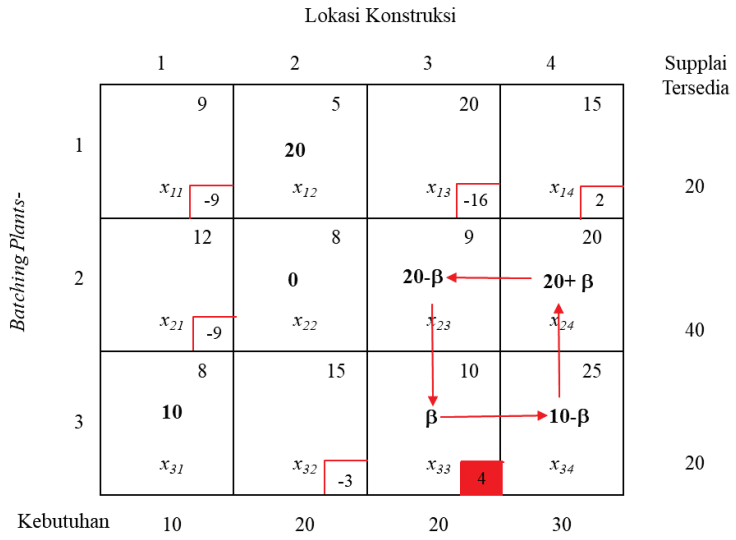
Tabel 5.8 Evaluasi variabel non basis tahap 2.

Variabel non basis	$U_i+V_j- c_{ij}$
x_{11}	$U_1+V_1-C_{11}$=0+0-9= -9
x_{13}	$U_1+V_3-C_{13}$= 0+4 -20= -16
x_{14}	$U_1+V_4-C_{14}$= 0+17-15=2
x_{21}	$U_2+V_1-C_{21}$=3+0-12= -9
x_{32}	$U_3+V_2-C_{32}$=8+5-15= -3
x_{33}	$U_3+V_3-C_{33}$=8+6-10= 4

Tabel 5.9 Rekapitulasi nilai *z* setiap variabel tahap 2.

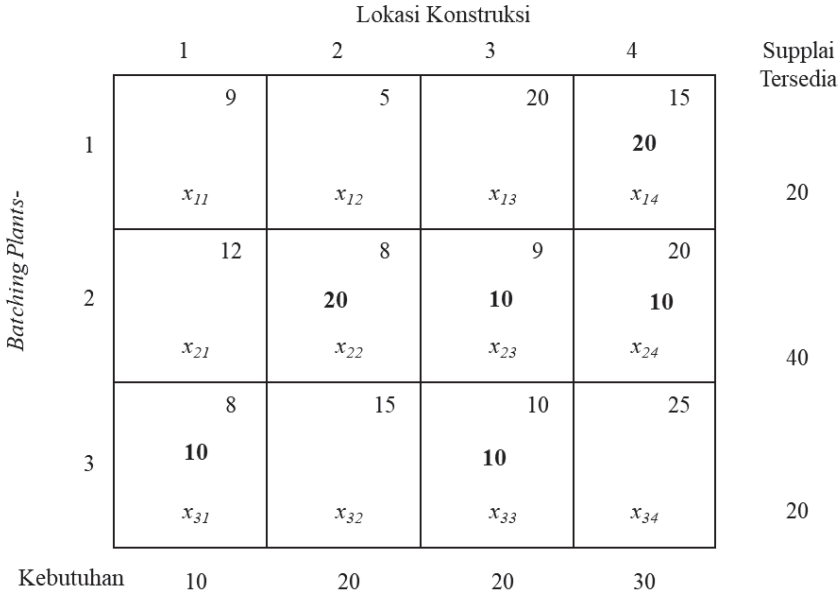
Basis	x_{11}	x_{12}	x_{13}	x_{14}	x_{21}	x_{22}	x_{23}	x_{24}	x_{31}	x_{32}	x_{33}	x_{34}
Z	-9	0	-16	2	-9	0	0	0	0	-3	4	0

Berdasarkan hasil evaluasi di atas, maka masih didapatkan variabel non basis yang bernilai positif. Dengan demikian, variabel non basis x_{33} tersebut akan dimasukkan ke dalam variabel basis. Selanjutnya akan dilakukan kembali *reduction process* dengan membentuk sebagai *loop* yang dimulai dan diakhiri dari variabel non basis yang akan masuk, yakni x_{14}. Proses *loop* hanya mengandung koneksi berupa horizontal dan vertikal saja dengan aturan tanda $+ \rightarrow - \rightarrow + \rightarrow - \rightarrow +$. Proses ini dapat dilihat pada Gambar 5.15. Nilai β dimasukkan pertama pada x_{33} dan proses *looping* disesuaikan dengan langkah yang sama seperti sebelumnya.



Gambar 5.15 *Reduction process* dengan nilai $\boldsymbol{\beta}$ tahap 2.

Proses selanjutnya dilakukan sama dengan tahapan sebelumnya, yaitu dengan mengalokasikan β sebanyak mungkin. Setelah itu dievaluasi kembali apakah masih terdapat nilai z yang positif pada variabel non basis pada hasil terakhir yang didapat. Langkah evaluasi sama seperti pada tahapan sebelumnya, yakni dengan menentukan nilai U_i dan V_j dari variabel basis yang ada.



Gambar 5.16 Nilai matriks pada tahap akhir perhitungan soal 5.1.

Tabel 5.10 Rekapitulasi nilai *z* setiap variabel tahap akhir.

Basis	x_{11}	x_{12}	x_{13}	x_{14}	x_{21}	x_{22}	x_{23}	x_{24}	x_{31}	x_{32}	x_{33}	x_{34}
Z	-5	-2	-14	0	-5	0	0	0	0	-6	0	-15

Setelah melakukan iterasi beberapa kali (proses lengkap tidak disampaikan), dapat dilihat pada Tabel 5.10 variabel non basis sudah tidak ada yang positif. Atas dasar ini dapat disimpulkan bahwa solusi telah optimum. Dengan demikian, jumlah produk yang perlu diangkut dari setiap *batching plant* menuju lokasi konstruksi dapat disesuaikan dari hasil konfigurasi terakhir dan ditabelkan pada Tabel 5.11 dengan total biaya sebesar 930. Nilai ini sesungguhnya sama dengan nilai yang diberikan oleh VAM.

Tabel 5.11 Rekapitulasi hasil *Transportation Problem.*

Dari *Batching Plant*-	Menuju Lokasi-	Jumlah Produk
1	4	20
2	2	20
2	3	10
2	4	10
3	1	10
3	3	10

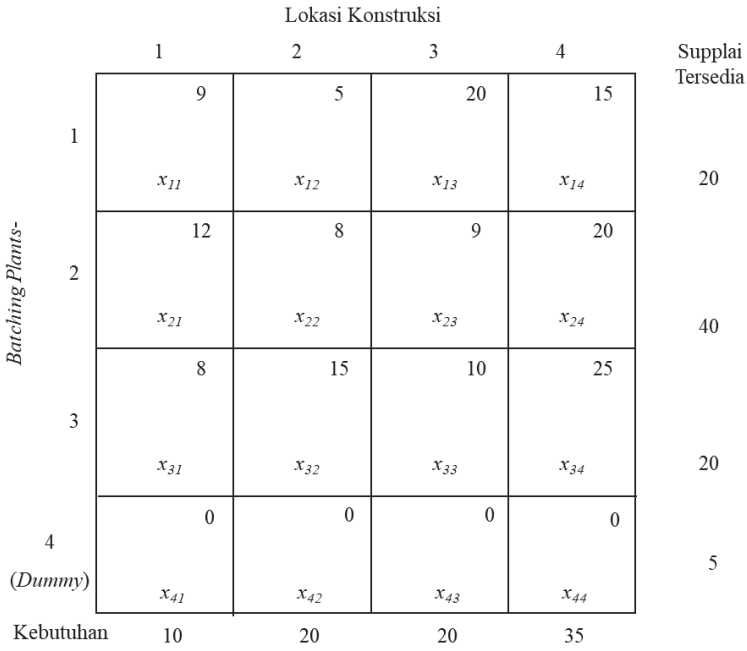
5.3. *Unbalanced Transportation Problem*

Pada bagian sebelumnya, permasalahan transportasi diasumsikan bahwa sumber memiliki jumlah persediaan yang sama persis dengan kebutuhan yang dibutuhkan oleh lokasi tujuan. Meskipun pada kenyataannya kondisi ini tidak pernah terjadi, yang mungkin terjadi adalah:

- Jumlah barang yang dimiliki oleh sumber lebih besar dibandingkan yang dibutuhkan oleh lokasi tujuan.
- Jumlah barang yang dimiliki oleh sumber lebih kecil dibandingkan dengan yang dibutuhkan oleh lokasi tujuan.

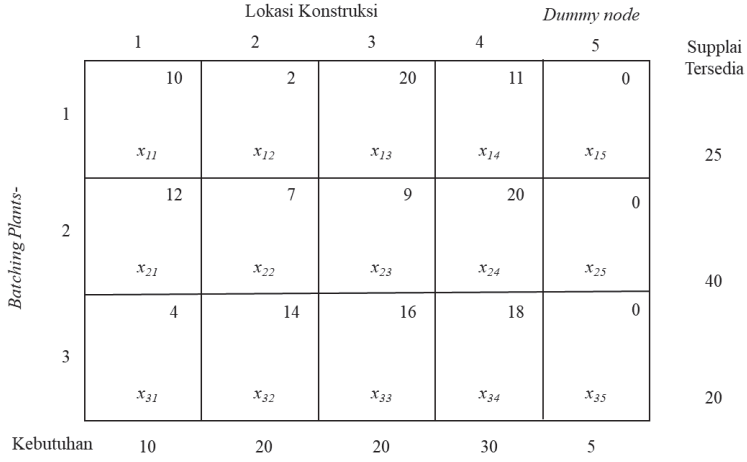
Oleh karenanya, jika kedua kasus ini terjadi model transportasi berubah menjadi *Unbalanced Transportation Problem*. Untuk menyelesaikan permasalahan ini, pada prinsipnya dapat diselesaikan dengan menggunakan prosedur yang dilakukan untuk menyelesaikan pada kasus *balanced*. Meskipun pada tahap awal, *supply* dari asal (ataupun kebutuhan di tujuan) harus diseimbangkan terlebih dahulu dengan menambahkan *dummy node* pada lokasi asal atau tujuan.

Sebagai contoh pada permasalahan yang sebelumnya dibahas, anggaplah kebutuhan dari lokasi tujuan 4 bertambah semula 30 berubah menjadi 35. Kondisi ini menyebabkan permasalahan ini menjadi tidak seimbang, dimana jumlah kebutuhan lebih besar daripada jumlah yang tersedia. Untuk mengatasi permasalahan ini maka dapat ditambahkan satu *dummy node* pada lokasi sumber dengan biaya transportasi sama dengan nol, seperti terlihat pada Gambar di bawah ini.



Gambar 5.17 Contoh penerapan *dummy nodes* pada *unbalance transportation problem* dengan kebutuhan lebih besar daripada suplai tersedia.

Hal yang sama dapat dilakukan jika jumlah suplai yang tersedia lebih besar dibandingkan dengan jumlah kebutuhan yang ada. Sebagai contoh, anggaplah suplai pada *batching plant-3* naik menjadi 25 sehingga akan terjadi kelebihan sebesar 5. Sehingga *dummy node* akan ditambahkan pada kolom seperti terlihat pada gambar berikut.



Gambar 5.18 Contoh penerapan *dummy nodes* pada *unbalance transportation problem* dengan kebutuhan lebih kecil daripada suplai tersedia.

Setelah melakukan proses penyeimbangan suplai yang tersedia dan kebutuhan yang ada, proses perhitungan dapat dilakukan sama seperti untuk kasus *balanced transportation problem.*

Pembaca yang tertarik untuk melihat penjelasan dalam bentuk audio-visual terkait Bab ini dapat mengunjungi playlist youtube berikut:

https://bit.ly/Playlist-TransportationProblem

6. Assignment Problem

6.1. Pengantar

Assignment problem merupakan salah satu bentuk dari *transportation problem* di mana terdapat pekerja atau perusahaan yang merepresentasikan sumber atau persediaan, dan suatu pekerjaan yang merepresentasikan tujuan. Pada *assignment problem,* nilai barang yang distribusikan dari asal menuju tujuan harus bernilai 1 dengan suatu biaya pekerja *i* ke pekerjaan *j* yang dituliskan sebagai c_{ij}. Fungsi tujuan dari permasalahan ini adalah untuk menugaskan setiap pekerja kepada pekerjaan tertentu dengan biaya yang paling minimum. *Assignment problem* dapat diselesaikan menggunakan template *transportation model* pada umumnya. Namun untuk mempermudah dalam metode penyelesaiannya, Kuhn–Munkres (Kuhn, 1955; Munkres, 1957) mengusulkan metode penyelesaian yang saat ini umum disebut sebagai *Hungarian Method* . Bentuk persamaan dari *assignment problem* adalah sebagai berikut.

$$x_{ij} = \begin{cases} 1, \text{jika pekerja} - i \text{ ditugaskan ke} - j \\ 0, \text{sebaliknya} \end{cases} \tag{6.1}$$

$$\min z = \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} \tag{6.2}$$

$$\sum_{i=1}^{m} x_{ij=1,}\ i = 1,2,\dots,m \tag{6.3}$$

$$\sum_{j=1}^{n} x_{ij=1,}\ j = 1,2,\dots,n \tag{6.4}$$

Penyelesaian *assignment problem* dapat dipermudah menggunakan bentuk tabel matriks dari *worker* dan *job* yang dapat dilihat pada tabel berikut.

Pekerja \ Pekerjaan	1	2	. . .	*n*	
1	c_{11}	c_{12}	. . .	c_{1n}	**1**
2	c_{21}	c_{22}	. . .	c_{2n}	**1**
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
n	c_{n1}	c_{n2}	. . .	c_{nm}	**1**
	1	**1**	. . .	**1**	

Gambar 6.1 Bentuk matriks untuk *Assignment Problem* (Taha, 2017).

6.2. Hungarian Method

Penyelesaian *assignment problem* menggunakan *hungarian method* dilakukan dengan prosedur berikut.

1. Identifikasi nilai baris terkecil dan kurangi dari seluruh baris yang ada.
2. Berdasarkan hasil yang sudah dikurangi nilai baris terkecil, identifikasi nilai kolom terkecil dan kurangi dari seluruh kolom yang ada.
3. Identifikasi solusi optimum dengan melihat entri 0 pada matriks yang didapatkan dari tahap sebelumnya. Jika jumlah garis minimum yang dibutuhkan untuk menutupi seluruh entri 0 sama dengan jumlah pekerjaan yang ada, maka solusi telah dianggap optimum. Jika tidak lakukan langkah 1.

Untuk memberikan ilustrasi lebih baik, berikut disampaikan contoh dari penerapan *Hungarian Method* menggunakan contoh soal 6.1 dan 6.2

Contoh Soal 6.1

Seorang pemilik proyek jalan membuka lelang untuk 3 paket pekerjaan yang berbeda (Paket 1, 2, dan 3). Lelang ini diikuti oleh 3 perusahaan (A, B, dan C) yang menawarkan setiap pekerjaan, dengan biaya yang dapat dilihat pada Tabel 6.1. Untuk meningkatkan kualitas pekerjaan, pemilik proyek menetapkan bahwa setiap perusahaan hanya bisa memenangkan satu paket pekerjaan saja. Apa rekomendasi yang dapat diberikan kepada pemilik proyek konstruksi dalam memilih setiap pemenang demi meminimalkan biaya untuk melaksanakan seluruh paket pekerjaan yang ada?

Tabel 6.1 Biaya penawaran ke pekerjaan (dalam Milyar Rp.).

	1	2	3
A	8	9	12
B	6	7	11
C	7	6	11

Tahap 1 Identifikasi nilai terkecil dari setiap baris

Nilai terkecil dari setiap baris diidentifikasikan pada Tabel 6.2, kemudian nilai tersebut menjadi pengurang pada setiap entri yang terlihat pada Tabel 6.3.

Tabel 6.2 Identifikasi nilai terkecil dari setiap baris.

	Paket 1	**Paket 2**	**Paket 3**	**Baris terkecil**
Perusahaan A	8	9	12	$b_1 = 8$
Perusahaan B	6	7	11	$b_2 = 6$
Perusahaan C	7	6	11	$b_3 = 6$

$$\sum_i b_i = 20 \tag{6.5}$$

Tabel 6.3 Pengurangan setiap baris dengan nilai baris terkecil.

	Paket 1	Paket 2	Paket 3	Baris terkecil
Perusahaan A	8-8=0	9-8=1	12-8=4	-
Perusahaan B	6-6=0	7-6=1	11-6=5	-
Perusahaan C	7-6=1	6-6=0	11-6=5	-
Kolom Terkecil	$k_1 = 0$	$k_2 = 0$	$k_3 = 4$	

Tahap 2 Identifikasi nilai terkecil dari setiap kolom

Tahap selanjutnya adalah dengan mengidentifikasikan nilai terkecil dari setiap kolom berdasarkan hasil dari Tabel 6.3 . Nilai dari setiap kolom akan dikurangi dengan nilai terkecil pada kolom tersebut yang ditinjau dan dapat dilihat pada Tabel 6.4.

$$\sum_i k_j = 4 \tag{6.6}$$

Tabel 6.4 Pengurangan setiap kolom dengan nilai kolom terkecil.

	Paket 1	Paket 2	Paket 3
Perusahaan A	0-0=0	1-0=1	4-4=0
Perusahaan B	0-0=0	1-0=1	5-4=1
Perusahaan C	1-0=1	0-0=0	5-4=1

Tahap 3 Identifikasi solusi optimum

Tahapan ini dilakukan dengan menggambar jumlah garis horizontal dan atau vertikal minimum yang mampu untuk menutupi nilai 0 pada Tabel 6.5. Jika jumlah garis yang dibutuhkan untuk menutupi entri bernilai 0 sama dengan jumlah pekerjaan yang tersedia, maka solusi optimum sudah ditemukan. Pada kasus soal ini, garis yang dapat dibuat untuk menutupi seluruh nilai 0 dapat dilihat pada Tabel 6.5. Jumlah garis yang dibutuhkan adalah sebanyak 3, sehingga sudah sesuai dengan jumlah pekerjaan yang tersedia, maka solusi optimum sudah tercapai.

Tabel 6.5 Penambahan garis untuk menutupi nilai 0.

	Paket 1	Paket 2	Paket 3
Perusahaan A	0	1	0
Perusahaan B	0	1	1
Perusahaan C	1	0	1

Berdasarkan tabel tersebut, maka bisa diambil nilai atau biaya terkecil dari setiap pekerjaan. Pada paket pekerjaan 3, biaya terkecil ada pada pada perusahaan A, pada pekerjaan 2, biaya terkecil ada pada perusahaan C, dan pada pekerjaan 3, biaya terkecil ada pada perusahaan A dan B, namun karena perusahaan A sudah mendapatkan pekerjaan maka pekerjaan 1 dapat diberikan pada perusahaan B. Hasil dari analisis tersebut dirangkum sebagai berikut.

- Paket pekerjaan 1 akan dikerjakan oleh perusahaan B.
- Paket pekerjaan 2 akan dikerjakan oleh perusahaan C.
- Paket pekerjaan 3 akan dikerjakan oleh perusahaan A.
- Total biaya yang dibutuhkan adalah: $6 + 6 + 12 = 24$

Total biaya juga dapat dihitung berdasarkan total b dan k yang sudah dihitung sebelumnya.

$$z = \sum_i b_i + \sum_i k_i = 24 \tag{6.7}$$

Contoh Soal 6.2

Dalam kondisi bencana, pusat bencana dihadapkan kepada permasalahan terkait mengembalikan fungsi 4 infrastruktur jalan (Ruas Jalan 1, 2, 3, 4) yang hancur karena gempa. Pusat bencana memiliki 4 tim alat berat untuk melakukan perbaikan infrastruktur jalan tersebut dengan waktu untuk menempuh dari lokasi tim ke ruas jalan yang rusak dapat dilihat pada tabel di bawah ini. Berikan rekomendasi kepada pusat bencana untuk menugaskan tim tersebut dalam rangka meminimalkan waktu perbaikan, anggaplah kondisi kerusakan dari keempat ruas jalan adalah seragam.

Tabel 6.6 Waktu perjalanan tim ke lokasi ruas yang rusak.

	Ruas 1	**Ruas 2**	**Ruas 3**	**Ruas 4**
Tim A	11	14	7	12
Tim B	10	10	11	12
Tim C	9	13	5	10
Tim D	7	10	6	9

Tahap 1 Identifikasi nilai terkecil dari setiap baris

Nilai terkecil dari setiap baris diidentikasikan dan direkap pada Tabel 6.8 dan selanjutnya setiap nilai pada baris akan dikurangi dengan nilai baris terkecil tersebut.

Tabel 6.7 Identifikasi baris terkecil (soal 6.2).

	Ruas 1	Ruas 2	Ruas 3	Ruas 4	Baris terkecil
Tim A	11	14	7	12	$b_1 = 7$
Tim B	10	10	11	12	$b_2 = 10$
Tim C	9	13	5	10	$b_3 = 5$
Tim D	7	10	6	9	$b_4 = 6$

$$\sum_{i=1}^{4} b_i = 28 \tag{6.8}$$

Tahap 2 Identifikasi nilai terkecil dari setiap kolom

Tahap selanjutnya adalah dengan mengidentifikasikan nilai terkecil dari setiap kolom berdasarkan hasil pengurangan baris sebelumnya dan direkap pada Tabel 6.8. Nilai dari setiap kolom akan dikurangi dengan nilai terkecil pada kolom tersebut. Hasil pengurangan dapat dilihat pada Tabel 6.9.

Tabel 6.8 Identifikasi kolom terkecil (soal 6.2).

	Ruas 1	Ruas 2	Ruas 3	Ruas 4	Baris terkecil
Tim A	11-7=4	14-7=7	7-7=0	12-7=5	-
Tim B	10-10=0	10-10=0	11-10=1	12-10=2	-
Tim C	9-5=4	13-5=8	5-5=0	10-5=5	-
Tim D	7-6=1	10-6=4	6-6=0	9-6=3	-
Kolom terkecil	$k_1 = 0$	$k_2 = 0$	$k_3 = 0$	$k_4 = 2$	

$$\sum_{j=1}^{4} k_j = 2 \tag{6.9}$$

Tabel 6.9 Hasil pengurangan setiap nilai dengan nilai terkecil baris dan kolom.

	Ruas 1	Ruas 2	Ruas 3	Ruas 4
Tim A	4-0=4	7-0=7	0-0=0	5-2=3
Tim B	0-0=0	0-0=0	1-0=1	2-2=0
Tim C	4-0=4	8-0=8	0-0=0	5-2=3
Tim D	1-0=1	4-0=4	0-0=0	3-2=1

Tahap 3 Identifikasi solusi optimum

Tahapan ini dilakukan dengan menggambar jumlah garis horizontal dan atau vertikal minimum yang mampu untuk menutupi nilai 0 pada Gambar 6.2 Jika jumlah garis yang dibutuhkan untuk menutupi nilai 0 sama dengan jumlah pekerjaan yang tersedia, maka solusi optimum sudah ditemukan. Pada kasus soal ini, garis yang dapat dibuat untuk menutupi seluruh nilai 0 dapat dilihat pada Gambar 6.2. Jumlah garis yang dibutuhkan adalah sebanyak 2, namun jumlah pekerjaan yang tersedia adalah 4, sehingga solusi optimum belum ditemukan.

	Ruas 1	Ruas 2	Ruas 3	Ruas 4
Tim A	4	7	0	3
Tim B	0	0	1	0
Tim C	4	8	0	3
Tim D	1	4	0	1

Gambar 6.2 Penambahan garis untuk menutup nilai 0.

Tahap 4 jika solusi belum optimal, maka tahapan selanjutnya dapat dilakukan dengan beberapa prosedur berikut.

- Menggambar jumlah garis horizontal dan atau vertikal minimum yang mampu untuk menutupi nilai 0 seperti yang sudah dilakukan pada Gambar 6.2.
- Memilih nilai terkecil dari *uncovered entry* (bagian yang tidak terkena oleh garis yang menutupi nilai 0), lalu dikurangi ke setiap nilai *uncovered entry,* dan ditambahkan ke setiap *intersection* dari kedua garis. Tahap ini dapat dilihat pada Gambar 6.3.
- Identifikasikan solusi optimum berdasarkan jumlah garis dan jumlah pekerjaan yang tersedia, apabila belum memenuhi, maka tahapan sebelumnya dilakukan kembali hingga solusi optimum tercapai.

	Ruas 1	Ruas 2	Ruas 3	Ruas 4
Tim A	4	7	0	3
Tim B	0	0	△	0
Tim C	4	8	0	3
Tim D	①	4	0	①

△ : pertemuan dua garis
▭ : entri yang tidak tertutupi
○ : entri dengan nilai terkecil yang tidak tertutupi

Gambar 6.3 Identifikasi *uncovered entries.*

Hasil pengurangan dari nilai *uncovered entry* terkecil ke setiap nilai *uncovered entry* dan penjumlahan ke setiap *intersection* dapat dilihat pada Gambar 6.4.

	Ruas 1	Ruas 2	Ruas 3	Ruas 4
Tim A	3	6	0	2
Tim B	0	0	2	0
Tim C	3	7	0	2
Tim D	0	3	0	0

Gambar 6.4 Pengurangan ke setiap *uncovered entry* dan penjumlahan ke *intersection.*

Tahapan selanjutnya adalah dengan menggambar jumlah garis horizontal dan atau vertikal minimum yang mampu untuk menutupi nilai 0 pada Gambar 6.4. Dari gambar tersebut terlihat bahwa jumlah garis minimum yang dibutuhkan adalah 3, belum sesuai dengan jumlah pekerjaan. Sehingga perlu dilakukan proses pada Tahap 4 kembali hingga semua nol dapat ditutupi dengan garis sejumlah 4. Setelah dilakukan iterasi berikutnya, diperoleh jumlah garis yang dibutuhkan adalah sebanyak 4 dan sudah sama dengan jumlah ruas yang harus diperbaiki (lihat Gambar 6.5), maka solusi optimum sudah tercapai pada tahap ini.

	Ruas 1	Ruas 2	Ruas 3	Ruas 4
Tim A	1	4	0	0
Tim B	0	0	4	0
Tim C	1	5	0	0
Tim D	0	3	0	0

Gambar 6.5 Penambahan garis untuk menutup nilai 0 pada iterasi akhir.

Berdasarkan gambar tersebut, maka bisa diambil nilai atau biaya terkecil dari setiap pekerjaan. Penentuan setiap pekerjaan dapat dimulai dari nilai terkecil (i.e., 0) yang berjumlah 1 pada pekerjaan tersebut, yakni dari Ruas 2, yang diberikan kepada Tim B. Ruas 1, nilai terkecil terdapat pada Tim B dan D, namun karena Tim B sudah mengerjakan Ruas 2, maka diberikan kepada Tim D. Pada Ruas 3 dari nilai terkecil, hanya Tim A yang belum menerima pekerjaan perbaikan, sehingga Ruas 3 diberikan kepada Tim A. Ruas 4 kemudian diberikan kepada Tim C yang tersisa. Hasil dari analisis tersebut dirangkum sebagai berikut.

- Ruas 1 akan dikerjakan oleh Tim D.
- Ruas 2 akan dikerjakan oleh Tim B.
- Ruas 3 akan dikerjakan oleh Tim A.
- Ruas 4 akan dikerjakan oleh Tim C.
- Total waktu tempuh dari lokasi tim ke jalan rusak adalah: $7 + 10 + 7 + 10 = 24$

Pembaca yang tertarik untuk melihat penjelasan dalam bentuk audio-visual terkait Bab ini dapat mengunjungi playlist youtube berikut:

https://bit.ly/Playlist-AssignmentProblem

7. Optimasi dengan *Metode Heuristic*

Dalam kerangka permasalahan optimasi, *heuristik* dan pengembangannya (e.g., *metahueristik*) memainkan peranan penting sebagai bagian dari teknik untuk menemukan solusi. Sejarah mencatat bahwa metode ini berkembang sejak tahun 1950-an dan mengemuka pada tahun 1960-an (Romanycia, 1985). Akar katanya pun di intrepretasikan pada berbagai definisi, ada yang mengganggap *heuristik* berasal dari Bahasa Yunani "*heuriskein*" yang artinya mencari, ada pula yang menganggap berasal dari akar kata "*heuretic*" yang merupakan cabang dalam ilmu logika sebagai landasan dalam mencari atau menemukan. Kata *heuristik* ini pun dapat diartikan sebagai kata sifat, kata benda hingga kata kerja.

Meskipun pada buku ini, dapat diartikan secara sederhana bahwa *heuristik* sebagai metode yang berisi kumpulan prosedur untuk mencari solusi dari sebuah masalah optimasi. Berbeda dengan pendekatan yang bersifat eksak (global optimal), pada metode ini tidak diharuskan untuk membuktikan bahwa solusi yang dihasilkan adalah global optimal. Akan tetapi cukup diuji apakah prosedur yang diterapkan dapat menghasilkan solusi yang secara umum dapat diterima. Oleh karenanya, metode ini tidak mengaransi adanya global optimal, sehingga *heuristik* dalam kerangka permasalahan optimasi memiliki beberapa peran berikut:

- Menghasilkan solusi awal sebagai input kepada teknik solusi yang bersifat eksak maupun *metaheuristik*.
- Memperbaiki kinerja dari metode *metaheuristik* atau *heuristik* lainnya.
- Membantu pencarian lokal dari solusi yang diberikan oleh metode eksak atau *metaheuristik* dalam rangka mempercepat penemuan solusi optimal.

Terdapat begitu banyak varian dari metode *heuristik*, meskipun dalam buku ini akan dibahas dua tipe algoritma saja, yaitu *greedy heuristics* dan *local search* yang umum diaplikasikan dalam bidang transportasi.

7.1. *Greedy Heuristics*

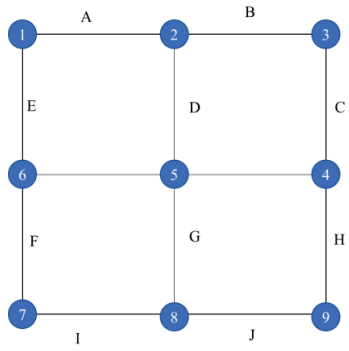
Greedy Heuristics atau disebut juga dengan *Greedy Algorithm* merupakan metode berbasis *heuristik* yang mengkonstruksi permasalahan optimasi ke dalam beberapa langkah atau objek untuk kemudian diselesaikan di setiap langkahnya. Metode ini tidak

mempertimbangkan apakah solusi optimal pada satu tahapan akan berkontribusi kepada solusi optimal untuk keseluruhan tahapan. Oleh karenanya metode ini terkadang terjebak di dalam lokal optimal, meskipun proses perhitungan dilakukan sangat mudah dan cepat jika dibandingkan metode berbasis solusi eksak maupun *metaheuristik*. Untuk lebih mudah dalam memahami prosedur dalam *greedy* ini, buku ini akan mencontohkan aplikasi *greedy* untuk menyelesaikan permasalahan optimasi sederhana terkait pemeliharaan jaringan jalan.

Contoh Soal 7.1

Anggaplah terdapat jaringan jalan seperti gambar dibawah ini (lihat Gambar 7.1) yang akan ditentukan jenis pemeliharaannya pada tahun kedepan. Terdapat tiga jenis pemeliharaan jalan yang dapat dilakukan yaitu, pemeliharaan rutin, pelapisan ulang, dan rehabilitasi. Jika diketahui arus lalu lintas dan nilai IRI pada masing-masing ruas, tentukan jenis pemeliharaan pada masing-masing ruas. Diketahui pula bahwa *budget* yang tersedia setiap tahun adalah Rp. 500 juta. Diketahui pula bahwa hubungan kondisi jalan $IRI^0_{t+1,j}$ terhadap arus lalu lintas n_{jt} pada suatu segmen-*j* pada waktu *t* adalah sebagai berikut:

$$IRI^0_{t+1,j} = [(1.033)IRI^0_{t,j}] + [(0.087805)n_{jt}] \quad (7.1)$$



Gambar 7.1 Jaringan jalan sederhana.

Tabel 7.1 Beban lalu lintas dengan nilai iri pada kondisi awal.

Nama Ruas	Beban Lalu Lintas (dalam ESAL)	Nilai awal IRI
A	129	6
B	191	3
C	201	6
D	228	4
E	486	12
F	310	5
G	384	4
H	461	6
I	309	3
J	262	11

Selain itu, diketahui bahwa jika dilakukan pemeliharaan maka nilai IRI akan berkurang mengikuti pola berikut:

Tabel 7.2 Performa akibat pemeliharaan.

Jenis Penanganan	*Performance Jump*	Biaya (juta Rp. per segmen)
Rehabilitasi ($i=1,\ \delta_j^1=1$)	$IRI^1_{(t,j)}$=2	100
Overlay ($i=2,\ \delta_j^2=1$)	$IRI^2_{(t,j)}=IRI_{(t-1,j)}$-4	50
Rutin ($i=3,\ \delta_j^3=1$)	$IRI^3_{(t,j)}=IRI_{(t-1,j)}$-1	10

Permasalahan optimasi ini akan diselesaikan dengan menggunakan pendekatan *greedy heuristik* yang berusaha membagi permasalahan menjadi beberapa tahapan dan mencari solusi optimal pada setiap tahapannya. Meskipun sebelum membahas lebih jauh implementasinya, maka penting untuk disusun terlebih dahulu fitur utama permasalahan optimasinya, yang terdiri i) fungsi objektif, ii) variabel keputusan, iii) fungsi kendala.

i) Fungsi objektif:

Terdapat beberapa pilihan di antaranya, meminimalkan nilai IRI, memaksimalkan selisih IRI tanpa penanganan dan IRI dengan penanganan, memaksimalkan rasio dari (selisih IRI tanpa penanganan dan IRI dengan penangangan) dan biaya penanganan. Pada buku ini anggap fungsi objektif terpilih adalah memaksimalkan selisih IRI dengan dan tanpa penanganan (lihat Persamaan (7.2)).

$$\max\ f=\sum_{j=1}^{J}(IRI^0_{t,j}-IRI^i_{t,j}),\forall t\in T, i=(1,2,3) \tag{7.2}$$

ii) Variabel keputusan:

Variabel keputusan adalah jenis penanganan pada setiap tahun pada masing-masing ruas $x^i_{(t,j)}$. Dimana hubungan antara IRI dan *variable* keputusan ini dapat dilihat pada Persamaan (7.3)

$$IRI^i_{t,j}=\sum_{i=1}^{3}x^i_{(t,j)}\delta^i_j\ \ \forall x^i_{(t,j)}\in\{0,1\} \tag{7.3}$$

iii) Fungsi kendala:

Fungsi kendala pada permasalahan ini adalah adanya batasan anggaran yang tersedia setiap tahun (Persamaan (7.4)). Serta harus dipastikan bahwa setiap ruas hanya boleh ditangani oleh satu jenis penanganan (Persamaan (7.5)).

$$\sum_{i=1}^{3} x_{(t,j)}^{i} c_j^i \leqslant Budget \forall x_{(t,j)}^{i} \in \{0,1\} \tag{7.4}$$

$$\sum_{i=1}^{3} x_{(t,j)}^{i} = 1 \tag{7.5}$$

Permasalahan ini kemudian dibagi berdasarkan tahun penanangannya, dimana di setiap tahun *greedy* akan memberikan solusi paling optimal.

Tahap Inisialisasi:

Tetapkan *t*=1

Tahap 1:

Tentukan nilai *fitness* pada masing-masing jenis penanganan pada setiap ruas. Nilai tersebut dapat diresumekan sebagai berikut:

Tabel 7.3 Nilai fungsi tujuan pada masing-masing penanganan di setiap ruas.

Nama Ruas	Beban Lalu Lintas (dalam ESAL)	Inisial IRI	Nilai IRI *)				Nilai Fungsi Objektif (*fitness*)		
			Tanpa Penangangan	Rehabilitasi	Overlay	Rutin	Rehabilitasi	Overlay	Rutin
A	129	6	7,33	2,00	2,00	6,33	5,33	5,33	1,00
B	191	3	4,78	2,00	2,00	3,78	2,78	2,78	1,00
C	201	6	7,96	2,00	2,00	6,96	5,96	5,96	1,00
D	228	4	6,13	2,00	2,00	5,13	4,13	4,13	1,00
E	486	12	16,66	2,00	8,00	15,66	14,66	8,66	1,00
F	310	5	7,89	2,00	2,00	6,89	5,89	5,89	1,00
G	384	4	7,50	2,00	2,00	6,50	5,50	5,50	1,00
H	461	6	10,25	2,00	2,00	9,25	8,25	8,25	1,00
I	309	3	5,81	2,00	2,00	4,81	3,81	3,81	1,00
J	262	11	13,66	2,00	7,00	12,66	11,66	6,66	1,00

*) Nilai IRI tidak boleh kurang dari 2.

Tahap 2:

Kumpulkan nilai *fitness*, berikan biaya yang dibutuhkan, serta berikan kode untuk memudahkan proses perhitungan (i.e., Nama Ruas-Nomor Jenis Penanganan)

Tabel 7.4 Nilai fungsi tujuan dan kode ruas.

Nilai *fitness*	Biaya	Kode Ruas
5,33	100	A1
2,78	100	B1
5,96	100	C1
4,13	100	D1
14,66	100	E1
5,89	100	F1
5,50	100	G1
8,25	100	H1
3,81	100	I1
11,66	100	J1
5,33	50	A2
2,78	50	B2

Nilai *fitness*	Biaya	Kode Ruas
5,96	50	C2
4,13	50	D2
8,66	50	E2
5,89	50	F2
5,50	50	G2
8,25	50	H2
3,81	50	I2
6,66	50	J2
1,00	10	A3
1,00	10	B3
1,00	10	C3
1,00	10	D3
1,00	10	E3
1,00	10	F3
1,00	10	G3
1,00	10	H3
1,00	10	I3
1,00	10	J3

Tahap 3:

Urutkan nilai *fitness* dari terbesar hingga terkecil

Tabel 7.5 Urutan nilai *fitness* dan kode ruas.

Urutan	Nilai *fitness*	Biaya	Kode Ruas
1	14,66	100	E1
2	11,66	100	J1
3	8,66	50	E2
4	8,25	100	H1
5	8,25	50	H2
6	6,66	50	J2
7	5,96	100	C1
8	5,96	50	C2
9	5,89	100	F1
10	5,89	50	F2
11	5,50	100	G1
12	5,50	50	G2
13	5,33	100	A1
14	5,33	50	A2
15	4,13	100	D1
16	4,13	50	D2
17	3,81	100	I1
18	3,81	50	I2
19	2,78	100	B1
20	2,78	50	B2
21	1,00	10	A3
22	1,00	10	B3
23	1,00	10	C3
24	1,00	10	D3
25	1,00	10	E3

Urutan	Nilai *fitness*	Biaya	Kode Ruas
26	1,00	10	F3
27	1,00	10	G3
28	1,00	10	H3
29	1,00	10	I3
30	1,00	10	J3

Tahap 4:

Pilih jenis penanganan mulai dari urutan teratas sambil menghitung total biaya penanganan. Jika total biaya penanganan akibat dilakukan pemeliharaan lebih besar dari *budget* yang ada, lanjutkan ke urutan selanjutnya hingga urutan terakhir. Selain itu harus dipastikan bahwa setiap ruas hanya memiliki satu jenis penanganan. Sebagai contoh pada Tabel 7.5 urutan ke 3 adalah Ruas E dengan *overlay*, akan tetapi ruas E sudah ditangani pada urutan pertama. Sehingga urutan ketiga dilewatkan langsung ke urutan selanjutnya. Selain itu, ruas yang tidak tertangani maka masuk dianggap ruas tanpa penanganan. Resume dari proses ini adalah sebagai berikut:

Tabel 7.6 Penentuan urutan penanganan.

Urutan Penangangan	Nama Ruas	Jenis Penanganan	Total Biaya	Nilai IRI	Nilai Fungsi Objektif
1	Ruas E	Rehabilitasi	100	2,0	14,66
2	Ruas J	Rehabilitasi	200	2,0	11,66
3	Ruas H	Rehabilitasi	300	2,0	8,25
4	Ruas C	Rehabilitasi	400	2,0	5,96
5	Ruas F	Rehabilitasi	500	2,0	5,89
6	Ruas A	Tanpa Penangangan	500	7,3	0
7	Ruas B	Tanpa Penangangan	500	4,8	0
8	Ruas D	Tanpa Penangangan	500	6,1	0
9	Ruas G	Tanpa Penangangan	500	7,5	0
10	Ruas I	Tanpa Penangangan	500	5,8	0
Nilai fungsi objektif pada saat t					**46,42**

Tahap 5:

Tambahkan t=t+1, jika t > waktu analisis (misal 3 tahun) maka berhenti. Jika tidak lanjutkan seperti dimulai pada tahap 1, dengan menganti nilai IRI inisial dengan Nilai IRI (t) pada Tabel 7.6

Dapat terlihat bahwa *greedy* memberikan solusi optimal pada setiap tahapan, meskipun metode ini tidak mengevaluasi apakah solusi optimal ini juga merupakan bagian dari solusi optimal pada seluruh jangka waktu.

7.2. *Local Search*

Metode *heuristik* lain yang kerap diaplikasikan adalah *local search*, yang pada prinsipnya mengeksplorasi solusi yang berdekatan dengan solusi sebelumnya (i.e., *neighborhood solution*). Atas dasar tersebut, metode ini umumnya diterapkan untuk mengeksplorasi dan

memperbaiki solusi dari metode lainnya. Sehingga metode ini umumnya tidak diterapkan secara mandiri.

Sebagai ilustrasi dari penerapan *local search* ini, anggaplah seorang kurir akan mengirimkan barang dari Lokasi A menuju Lokasi E. Untuk menuju lokasi E dari A kurir harus menempuh lokasi B, C dan D. Adapun jarak dari masing-masing lokasi dapat diperoleh pada tabel berikut. Jika kurir ingin meminimalkan jarak perjalanan berikan rekomendasi kepada kurir tersebut.

Tabel 7.7 Jarak antar lokasi.

	A	B	C	D	E
A	0	1	2	2	1
B	1	0	2	1	2
C	2	1,5	0	2	2,5
D	2	1	2	0	1
E	1	2	2,5	1	0

Sebelum mengaplikasikan metode *local search,* formulasikan terlebih dahulu permasalahan di atas menjadi permasalahan optimasi:

- Fungsi tujuan: meminimalkan jarak tempuh
- Variabel keputusan: urutan kunjungan ke masing-masing lokasi
- Fungsi kendala: berawal di A berakhir di E, dan harus melewati B,C,D

Tahap 1:

Bangkitkan solusi awal, jika *local search* digunakan untuk memperbaiki solusi dari metode lain. Solusi awal merupakan hasil dari solusi metode lain tersebut. Anggaplah solusi awal adalah A-B-C-D-E

Tahap 2:

Evaluasi nilai *fitness* dari solusi yang dapat diresumekan sebagai berikut:

Tabel 7.8 Nilai *fitness* solusi awal.

Urutan	AB	BC	CD	DE	Total Jarak
ABCDE	1	2	2	1	6

Tahap 3:

Evaluasi solusi yang berdekatan dengan solusi sebelumnya. Misal tukar BC menjadi CB sehingga solusi menjadi ACBDE. Evaluasi nilai *fitness* nya seperti pada Tabel 7.9.

Tabel 7.9 Nilai objektif solusi yang berdekatan.

Urutan	AC	CB	BD	DE	Total Jarak
ACBDE	2	1,5	1	1	5,5

Tahap 4:

Lakukan tahapan seperti Tahap 3 hingga kriteria berhenti tercapai, misalnya dengan nilai *fitness* tidak lebih baik dari sebelumnya. Sebagai contoh, proses evaluasi akan dilakukan kembali dari solusi pada Tahap 3, sehingga menjadi ACDBE. Terlihat bahwa nilai fungsi objektif lebih buruk dibandingkan tahap sebelumnya, sehingga dianggap solusi sebelumnya merupakan solusi optimal.

Tabel 7.10 Nilai objektif solusi yang berdekatan.

Urutan	AC	CD	DB	BE	Total Jarak
ACDBE	2	2	1	2	7

Contoh di atas (lihat Tabel 7.10) memberikan ilustrasi implementasi dari *local search* yang mengevaluasi solusi di sekitar solusi yang ditemukan sebelumnya. Sehingga *local search* umumnya memberikan solusi yang bersifat lokal optimal. Meskipun jika digabungkan dengan metode lain, *local search* memberikan peluang perbaikan algoritma secara signifikan. Sebagai contoh jika digabungkan dengan *Genetic Algorithm*, metode *local search* memberikan dampak perbaikan performa yang cukup signifikan.

Pembaca yang tertarik untuk melihat penjelasan dalam bentuk audio-visual terkait Bab ini dapat mengunjungi playlist youtube berikut:

https://bit.ly/Playlist-Heuristik

8. Optimasi dengan Metode *Metahueristik*

8.1. Pengantar *Metahueristik*

Kata *metaheuristik* pada prinsipnya memiliki dua akar suku kata yaitu "*meta*" dan "*heuristic"*. Kata dasar *heuristik* berasal dari Bahasa Yunani heuriskein yang artinya "*to find*" atau berusaha mencari, kemudian meta merupakan sesuatu yang bersifat di atas atau tingkat atas, sehingga *metaheuristik* merupakan metode *heuristik* yang ditambahkan algoritma di atasnya untuk mendapatkan solusi yang lebih baik. *Metahueristik* merupakan salah satu jenis metode berbasis aproksimasi yang tidak mengaransi adanya global optimal. Meskipun dalam banyak kasus di bidang rekayasa transportasi, permasalahan optimasi bersifat kompleks yang menyebabkan metode berbasis eksak tidak dapat digunakan. Pada kasus dimana metode eksak tidak dapat digunakan, solusi berbasis *metaheuristik* umumnya digunakan untuk memecahkan permasalahan optimasi.

Silver (2004) memberikan alasan-alasan yang kuat mengapa metode *heuristik*, sama halnya dengan penggunaan *metaheuristik*, dipilih untuk menyelesaikan masalah optimasi.

1. Menyediakan pemahaman bagi pengguna tentang bagaimana aturan dalam algoritma dioperasikan, terutama bagaimana parameter kunci mempengaruhi aksi yang dipilih.
2. Menunjukkan peningkatan performa dibandingkan metode yang sudah ada saat ini.
3. Memberikan hasil yang lebih cepat dan dapat diterima, dibandingkan metode eksak.
4. Tidak terlalu sensitif terhadap karakterisik variasi dalam masalah dan kualitas data.
5. Dapat dikombinasikan dengan metode eksak untuk mendapatkan solusi eksak yang lebih cepat dibandingkan menggunakan metode eksak semata.

Metaheuristik sudah mempunyai sejarah yang panjang, yang mungkin ditandai pada tahun 1965 dengan kemunculan *First Evolution Strategy*. Metodologi ini menjadi dasar pengembangan *Genetic Algorithm* pada tahun 1975. Perkembangannya semakin tidak terbendung dengan kehadiran *Simulated Annealing* pada tahun 1983, *Tabu Search* pada tahun 1986, *Ant Colony Optimization* pada tahun 1991, *Particle Swarm Optimization* pada tahun 1996, hingga pada tahun 1997 terdapat *Variable Neighborhood Search*. Pengembangan metode berbasis *metaheuristik* ini masih berkembang hingga saat ini, sebagai contoh setelah

tahun 2000+ terdapat *Parallel and Distributed Computing in Metahueristics* maupun *Glowworm Swarm Optimization* (2005).

8.2. Karakteristik Masalah yang Sesuai dengan *Metahueristik*

Berbagai keunggulan dalam mengadaptasi serta mengimplementasikan metode *metaheuristik* membuat metode ini menjadi pilihan ketika menghadapi persoalan optimasi yang sulit dipecahkan. Kemampuan menghasilkan solusi optimal dalam waktu yang relatif lebih singkat menjadi keunggulan utama dari metode *metaheuristik*, terutama dalam masalah yang termasuk dalam kategori *Non-polynomial Problem* (NP). Seperti yang telah disinggung pada awal tulisan, waktu penyelesaian merupakan salah satu parameter utama dari kinerja suatu metode. Semakin cepat waktu yang dibutuhkan untuk menghasilkan solusi yang dapat diterima, maka semakin baik kinerja dari metode tersebut.

Oleh karena itu, menjadi sangat penting untuk mengetahui jenis dari masalah berdasarkan waktu yang dibutuhkan oleh metode eksak untuk memecahkan masalah optimasi. Secara sederhana, permasalahan ditinjau dari sisi waktu yang dibutuhkan, dapat dibagi menjadi dua kelompok besar. Kelompok pertama adalah kelompok masalah yang dapat diselesaikan oleh metode eksak dalam fungsi waktu yang mengikuti kurva polinomial atau dikenal dengan *Polynomial Problem* (P). Contoh umum dari permasalahan berjenis P ini adalah *shortest path problems*, *minimum spanning tree*, *sorting*, *string editing, maximum flow network* dll. Kelompok kedua dikategorikan sebagai kelompok *Non-polynomial Problem* (NP). Pada kelompok masalah kedua ini, metode eksak yang dikenal tidak mampu menyelesaikan masalah optimasi dalam waktu yang mengikuti kurva polinomial. Sehingga waktu penyelesaian permasalahan menggunakan metode eksak tidak mungkin diterima secara akal sehat (lihat Tabel 8.1). Oleh karena itu, metode pendekatan berperan besar pada karakter masalah ini, yang mampu menyelesaikan masalah tersebut dalam waktu yang mengikuti kurva polinomial. Untuk lebih memahami perbedaan NP dan P, dapat dilihat contoh sederhana berikut.

Tabel 8.1 Contoh waktu penyelesaian permasalahan NP dan P.

Jumlah variabel keputusan (x)	*Polynomial Problem*		*Non-Polynomial Problem*	
	Ukuran masalah $f(x) = 2x$	Waktu*) (menit)	Ukuran masalah $f(x) = 2^x$	Waktu*) (menit)
1	2	4	2	4
2	4	8	4	8
3	6	12	8	16
...	...	...	...	...
17	34	68	131072	262.144

*) waktu yang dibutuhkan untuk menyelesaikan satu permasalahan adalah 2 menit.

Asumsikan bahwa *x* merupakan variabel yang menentukan ukuran masalah. Berdasarkan tabel di atas, dengan mudah kita pahami bahwa peningkatan ukuran masalah pada NP akan berdampak signifikan pada ukuran masalah secara keseluruhan. Sebagai contoh, peningkatan jumlah variabel keputusan sebanyak 17 pada NP mampu meningkatkan ukuran masalah hingga 100,000 kali. Dengan asumsi bahwa 1 ukuran masalah membutuh waktu 2 menit, maka dibutuhkan waktu penyelesaian selama 262,144 menit atau 182 hari. Jika dibandingkan dengan ukuran masalah pada *P,* peningkatan variabel x tidak berdampak signifikan pada ukuran masalah fungsi *x*. Oleh karena itu, fungsi $f(x) = 2^x$ dapat dianggap berjenis NP, karena peningkatan variabel/ukuran masalah akan berdampak secara signifikan pada peningkatan waktu penyelesaian, dan tidak mengikuti kurva polinomial, sehingga sangat tidak rasional menyelesaikannya menggunakan metode eksak. Contoh umum permasalahan yang termasuk dalam NP adalah *traveling salesman problem* (TSP), *knapsack problem*, dan *vehicle routing problem (VRP)*. Berikut ini dijabarkan beberapa tipe permasalahan berjenis NP, yang umumnya membutuhkan metode penyelesaian berbasiskan pendekatan.

8.2.1 *Travelling Salesman Problem*

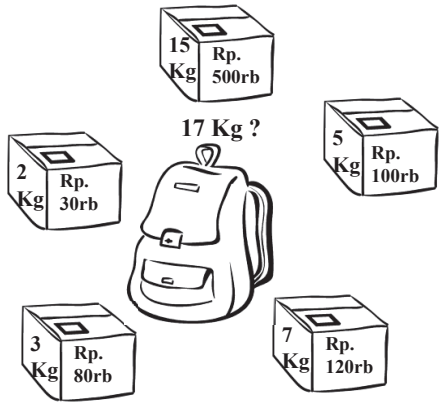
Meskipun sudah diformulasikan sejak tahun 1930an, pembahasan tentang TSP sepertinya tidak pernah hilang di forum-forum akademis. TSP merupakan masalah yang terdiri dari list kota atau depot beserta jarak antar kota/depotnya, dimana algoritma dipekerjakan untuk menemukan rute terpendek untuk mengunjungi setiap kota ataupun depot. TSP sering digunakan sebagai *benchmark* berbagai masalah optimasi, untuk membandingkan berbagai metode yang diusulkan dalam menyelesaikan masalah optimasi yang tergolong dalam kategori NP. Efek dari jumlah kota/depot yang harus dikunjungi terhadap ukuran masalah dapat dilihat pada Tabel 8.2 (Talbi, 2009).

Tabel 8.2 Efek jumlah kota pada ukuran ruang pencarian TSP.

Jumlah Kota	Ukuran Ruang Pencarian*)
5	120
10	3,628,800
75	2.5×10^{109}

*) total kombinasi urutan kota yang dikunjungi untuk meminimalkan jarak tempuh

(e.g., 1-2-3-4or 3-2-1-4 or 4-3-1-2 dsb).



Gambar 8.1 Ilustrasi *Knapsack Problem*.

8.2.2. *Knapsack Problem*

Contoh lain dari permasalahan berjenis NP adalah *Knapsack Problem*. Seperti TSP, permasalahan ini telah berkembang sejak seabad yang lalu dan dijadikan studi kasus sejak tahun 1897. *Knapsack Problem* mencakup proses optimasi sebuah atau beberapa tas/ransel yang memiliki kapasitas (e.g.,, berat atau volume) tertentu, yang dapat diisi oleh beberapa item yang memiliki berat/volume serta nilai tertentu. Permasalahan ini berusaha untuk memasukan item/barang ke dalam tas yang memiliki batasan kapasitas agar menghasilkan nilai yang maksimal. Permasalahan ini sesungguhnya refleksi atas permasalahan manusia yang umum, seperti mengemas pakaian ketika bepergian atau naik gunung. Manusia dituntut untuk menghasilkan manfaat yang maksimal dari sebuah koper atau ransel dengan membawa barang-barang yang memiliki nilai manfaat tertentu.

8.2.3. *Vehicle Routing Problem*

The vehicle routing problem (VRP) merupakan bagian dari permasalahan optimasi yang pertama kali dikembangkan oleh Dantzig and Ramser pada tahun 1959. VRP mencakup permasalahan optimasi rute yang berasal-berakhir di depot, untuk mengantar barang/orang dengan kebutuhan yang telah diketahui sebelumnya, dengan menggunakan biaya yang minimum maupun dengan jumlah kendaraan terkecil. Oleh karena itu, permasalahan ini setidaknya terdiri dari satu atau beberapa depot dan beberapa titik pelanggan (i.e., *demand*) yang terpisah satu sama lain, dimana *demand* pada titik pelanggan harus dibawa ke depot menggunakan sebuah kendaraan (contoh optimasi bus jemputan sekolah). Permasalahan ini dapat pula berupa permasalahan optimasi rute kendaraan yang berasal dari depot dengan tujuan mengirimkan barang kepada pelanggan untuk memenuhi

kebutuhannya. Menurut Silver (2004) setidaknya terdapat 3 jenis VRP berdasarkan jenis kendalanya (*constraints*).

1. Kendala kapasitas: kendaraan memiliki batasan kapasitas (e.g., jumlah unit) untuk membawa barang/orang.
2. Kendala jarak/waktu tempuh: kendaraan memiliki batasan jarak yang dapat ditempuh, maupun terdapat batasan waktu dalam mengantarkan barang/orang.
3. Kendala waktu jendela (i.e., *time window*): terdapat batasan waktu paling awal dan paling akhir dalam mengambil atau mengirimkan barang pada setiap titik pelanggan, maupun terdapat batasan waktu paling awal dan paling akhir kendaraan untuk mencapai tujuan akhir.

8.3. Perencanaan Transportasi menggunakan *Metaheuristik*

Penggunaan metode yang tepat dalam perencanaan transportasi dapat memberikan kesempatan bagi masyarakat luas untuk menikmati perjalanan secara lebih efisien. Akan tetapi, perencana transportasi selalu berhadapan dengan terbatasnya dana untuk mewujudkan infrastruktur transportasi, sementara kebutuhan dalam pembangunan infrastruktur transportasi sangatlah besar. Oleh karena itu, kemampuan dalam menyusun agenda prioritas menjadi sebuah keharusan bagi seorang perencana jaringan transportasi. Agenda prioritas pembangunan infrastruktur transportasi harus mampu menghasilkan strategi investasi yang optimal, dengan melakukan evaluasi kepada kebijakan-kebijakan yang mungkin diambil dalam perencanaan jaringan transportasi serta memilih infrastruktur transportasi yang memberikan kemanfaatan terbesar dalam *budget* yang terbatas. Pendekatan dalam penyusunan strategi ini umum dikenal sebagai bagian dari *network design problem* (NDP), yang mencakup permasalahan untuk meningkatkan performa jaringan transportasi (e.g., total waktu perjalanan, waktu tundaan) dengan melakukan peningkatan kapasitas infrastruktur transportasi yang ada, maupun dengan melakukan pembangunan infrastruktur baru.

Secara umum NDP terbagi menjadi 3 jenis, yaitu *continuous* NDP (CNDP), *discrete* NDP (DNDP) and *mixed* NDP (MNDP). CNDP biasanya digunakan untuk menentukan kapasitas optimal dari infrastruktur eksisting yang direpresentasikan dengan menggunakan variabel berjenis kontinu. CNDP telah diaplikasikan untuk mengatasi berbagai jenis masalah di bidang transportasi termasuk peningkatan kapasitas jalan (e.g., Abdulaal and LeBlanc, 1979; Chiou, 2005; Zhang and Gao, 2007), perbaikan sinyal lalu

lintas (e.g., LeBlanc and Boyce, 1986; Ziyou and Yifan, 2002; Chiou, 2008) and jalan tol (e.g., Yang, 1997). DNDP mencakup permasalahan dalam penambahan infrastruktur baru, perbaikan infrastruktur eksisting, yang direpresentasikan dengan bilangan integer (i.e., 0-1). Sementara itu, secara praktikal rekayasa jaringan transportasi mencakup variabel keputusan yang berbentuk diskret dan kontinu yang tidak dapat dipisahkan satu sama lain, MNDP menggabungkan variabel keputusan diskret dan kontinu secara simultan (e.g., Yang and Meng, 2001; Cantarella and Vitetta, 2006; Luathep *et al.*, 2011)

Meskipun pada buku ini contoh-contoh dan aplikasi akan lebih memfokuskan diri pada permasalahan transportasi berbasiskan DNDP. Pendekatan ini dianggap lebih mudah diaplikasikan dalam kerangka perbaikan performa dari infrastruktur transportasi dibandingkan dengan CNDP. Selain itu, Boyce and Janson (1980) juga berpendapat bahwa fungsi diskret (i.e., DNDP) lebih cocok dalam melakukan perencanaan transportasi, karena peningkatan jumlah lajur dari jalan dapat dilakukan dalam bilangan fraksional/kontinu. Meskipun DNDP memiliki kelemahan dalam hal waktu komputasi jika mencakup jumlah variabel keputusan (i.e., 0–1) yang besar, selain DNDP dikenal sebagai bagian dari optimasi kombinatorial yang sulit dipecahkan menggunakan metode eksak. Sehingga *metahueristik* memiliki peluang yang baik untuk diterapkan pada perencanaan transportasi untuk mengurangi waktu komputasi dengan tetap memberikan hasil yang dapat diterima.

Penelitian dalam DNDP untuk perencanaan transportasi setidaknya terfokus atas 3 hal, yaitu level strategis yang mencakup keputusan jangka panjang dalam pengembangan infrastruktur transportasi. Hal ini termasuk perencanaan transportasi untuk pembangunan jalan baru, jalan bebas hambatan, rel kereta api, pelabuhan, hingga peningkatan kapasitas infrastruktur eksisting (e.g., Steenbrink, 1974; Leblanc, 1975; Poorzahedy and Turnquist 1982; Chen and Alfa, 1991; Xiong and Schneider, 1992; Yang and Bell, 1998; Gao *et al.*, 2005; Poorzahedy and Abulghasemkai, 2005; Poorzahedy and Rouhani, 2007; Yamada *et al.*, 2009). Grup kedua bekerja dalam kerangka taktikal yang berfokus pada peningkatan efektifitas dari infrastruktur transportasi. Keputusan dalam level taktis mencakup penentuan orientasi jalan satu arah, pengalokasian lajur pada jalan dua arah, dan perubahan jalan dua arah menajadi jalan satu arah (e.g., Lee and Yang, 1994; Drezner and Wesolowsky, 1997; Drezner and Salhi, 2000; Drezner and Salhi, 2002; Zhang and Gao, 2007; Wu *et al.*, 2009; Long *et al.*, 2010). Fokus terakhir menggabungkan perencanaan level strategis dan taktis dalam satu studi (e.g., Drezner and Wesolowsky, 2003; Miandoabchi and Farahani, 2010).

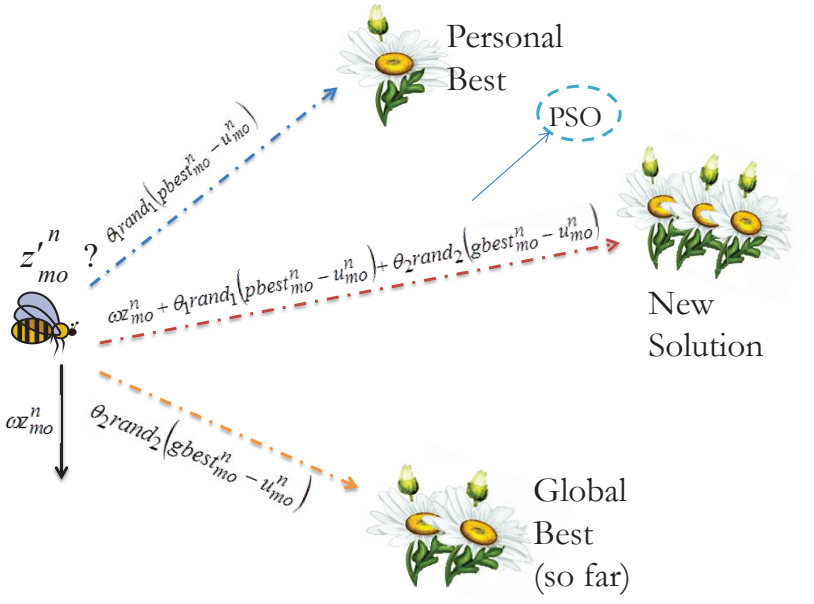
Metode untuk menyelesaikan DNDP pun telah mencapai peningkatan yang subtansial, diawali dengan pendekatan berbasis *BB algorithm* (i.e., LeBlanc, 1975), permasalahan ini pun telah menarik para peneliti untuk mengaplikasikan *metaheuristic* di bidang perencanaan transportasi. Drezner dan Salhi (2000) menyelesaikan permasalahan dalam rekayasa konfigurasi jalan satu dan dua arah dengan menggunakan tabu search (TS). Drezner and Wesolowsky (2003) kemudian mempresentasikan DNDP dengan mengaplikasikan genetik algoritma (GA), simulated anneling (SA) dan TS untuk menyelesaikannya. Ulasan komprehensif tentang pengembangan dari DNDP dapat dilihat pada Yang and Bell (1998) and Farahani *et al.* (2013).

Pembaca yang tertarik untuk melihat penjelasan dalam bentuk audio-visual terkait
Bab ini dapat mengunjungi playlist youtube berikut:
https://bit.ly/Playlist-Metaheuristics

9. *Particle Swarm Optimisation* (PSO)

9.1. Pendahuluan

PSO merupakan teknik komputasi evolusioner yang terinspirasi dari perilaku sekelompok burung atau lebah. Partikel yang merepresentasikan burung atau lebah, dapat dianggap sebagai kandidat solusi dari permasalahan optimasi yang terbang sepanjang ruang permasalahan untuk menemukan solusi optimal. Setiap partikel memiliki informasi tentang posisi dan nilai dari posisi tersebut yang diperoleh dengan mengevaluasi posisi terhadap fungsi tujuan. Setiap partikel memperbaharui posisi mereka berdasarkan vektor kecepatannya. Vektor kecepatan ditentukan berdasarkan posisi partikel itu sendiri dan posisi yang pernah dialami oleh partikel lain dalam satu kelompok. Secara khusus, setiap vektor kecepatan mencakup informasi tentang posisi terbaik yang pernah disinggahi oleh partikel (*Pbest*) dan posisi terbaik dalam kelompok (*Gbest*). Perhitungan vektor kecepatan melakukan kombinasi serta menyeimbangkan ketiga tendensi ini untuk memperoleh solusi seperti yang dapat dilihat pada Gambar 9.1.



Gambar 9.1 Mekanisme pembaharuan vektor kecepatan PSO.

Kennedy and Eberhart merupakan peneliti pertama yang merancang PSO sebagai metode optimasi untuk digunakan pada permasalahan berjenis bilangan riil. Kemudian pada tahun 1997, Eberhart dan Kennedy memperluas aplikasi PSO dengan mengembangkan PSO untuk mengatasi optimasi kombinatorial dalam bilangan biner (0-1), yang kemudian dikenal sebagai *discrete binary* PSO (DBPSO). PSO untuk bilangan biner ini telah menarik banyak peneliti untuk melakukan pengembangan lanjutan (e.g., Salman *et al*., 2002; Shen *et al*., 2004; Wang *et al*., 2008). Shen *et al*. (2004) mengembangkan *modified binary PSO* (MBPSO) dengan memperbaharui strategi untuk berganti posisi partikel. Menhas *et al*. (2012) kemudian mengaplikasikan *the probability-based discrete binary PSO* (PBPSO) untuk memecahkan permasalahan optimasi (lihat juga Menhas *et al*. 2011,2012; Wang *et al*. 2008 untuk PBPSO). Zukhruf *et al*. (2014) memodifikasi PBPSO dengan menambahkan strategi pembaharuan posisi partikel pada ekisiting PBPSO, dan menamakan metode ini sebagai *Modified PBPSO* (MPBPSO). Keempat metode ini akan dibahas pada buku ini untuk memberikan gambaran yang lebih luas tentang pengembangan algoritma PSO.

9.2. Tipe *Binary* PSO

9.2.1. *Discrete Binary* PSO (DBPSO)

DBPSO merupakan tipe pertama dari PSO yang dapat menangani permasalahan optimasi kombinatorial. Berbeda dengan PSO kontinu, dalam DBPSO, sebuah partikel bergerak dalam ruang pencarian yang dibatasi oleh satu atau nol pada setiap dimensi masalah. Ide dasar dari algoritma DBPSO dapat dijelaskan sebagai berikut:

$$z'^{n}_{mo} = \omega z^{n}_{mo} + \theta_1 rand_1(pbest^{n}_{mo} - u^{n}_{mo}) + \theta_2 rand_2(gbest^{n}_{mo} - u^{n}_{mo}) \quad (9.1)$$

$$Sig(z'^{n}_{mo}) = 1/(1 + e^{-z'^{n}_{mo}}) \quad (9.2)$$

$$u'^{n}_{mo} = \begin{cases} 1 \; if \; rand \leq Sig(z'^{n}_{mo}) \\ 0 \text{ otherwise} \end{cases} \quad (9.3)$$

Dimana,

z^{n}_{mo} : Kecepatan partikel *m* pada saat iterasi ke *n* pada *bit* ke-*o*,

z'^{n}_{mo} : Kecepatan partikel *m* pada saat iterasi ke *n*+1 pada *bit* ke-*o*,

u^{n}_{mo} : Posisi partikel *m* pada saat iterasi ke *n* pada *bit* ke-*o* (bilangan biner 0–1),

u'^{n}_{mo} : Posisi partikel *m* pada saat iterasi ke *n*+1 pada *bit* ke-*o* (bilangan biner 0–1),

$pbest^{n}_{mo}$: Posisi personal terbaik dari partikel *m* pada saat iterasi ke *n* pada *bit* ke-*o* (bilangan biner 0–1),

$gbest^{n}_{mo}$: Posisi terbaik global pada saat iterasi ke *n* pada *bit* ke-*o* (bilangan biner 0–1),

ω : bobot inersia,
$\theta_1\ \theta_2$: Faktor pembelajaran untuk *Pbest* dan *Gbest*,
$Sig(\ z_{mo}^n)$: Fungsi transformasi sigmoid pada partikel *m*,
$rand_1$,$rand_2$: Bilangan acak antara 0 dan 1.

z_{mo}^n menunjukan kecepatan dari partikel yang merepresentasikan probabilitas dari *bit* untuk bernilai satu. Sebagai contoh, jika z_{mo}^n=0,30, kemudian posisi partikel-*m* pada *bit-o* diiterasi *n* (i.e., u_{mo}^n) memiliki peluang 30% untuk menjadi 1 dan peluang 70% untuk menjadi nol. Setiap partikel menjaga informasi tentang posisi terbaik yang pernah partikel jumpai hingga saat ini yang ditunjukan sebagai $pbest_{mo}^n$, dan posisi terbaik di antara semua partikel di dalam populasi yang dinamakan sebagai $gbest_{mo}^n$. θ_1 dan θ_2 menunjukan faktor akselerasi dari setiap partikel untuk menuju posisi personal terbaik dan posisi global terbaik. Peran kedua parameter ini menjadi menjadi penting untuk mengontrol keseimbangan antara pembelajaran sosial dan personal dari partikel. Nilai θ_1 dan θ_2 dapat diatur sama dengan 2.

Selain itu, ω merupakan bobot inersia (Shi and Eberhart, 1998) yang berperan besar dalam menyeimbangkan pencarian global dan pencarian lokal. Bobot inersia yang besar akan memfasilitasi partikel untuk melakukan eksplorasi global, sementara bobot inersia yang kecil akan membuat partikel mengeksploitasi secara lokal. Kennedy and Eberhart (1997) juga menambahkan vel_{max} $(-vel_{min})$ untuk membatasi kecepatan dari partikel, dimana mereka mengatur vel_{max} $(-vel_{min})$ sebesar 6,0, and yang berdampak pada probabilitas fungsi sigmoid (i.e.,$Sig(\ z_{mo}^n)$) akan dibatasi pada rentang 0,0025 sampai 0,9975.

Adapun prosedur dari DBPSO dapat dilihat sebagai berikut (lihat, Menhas, 2012):

Langkah 1. Inisialisasi

- Atur nilai bobot inersia (i.e., ω), faktor pembelajaran θ_1 dan θ_2, jumlah partikel *N* dalam *swarm* dan jumlah iterasi.
- Tetapkan batas atas dan batas bawah kecepatan partikel (i.e., vel_{max} $(-vel_{min})$)
- Tetapkan kecepatan awal partikel , (dalam buku ini ditetapkan nol).

$$z_m^0 = [z_{m1}^0, z_{m2}^0, \dots, z_{mo}^0, \dots, z_{mD}^0]$$

- Hasilkan posisi awal dari partikel dengan menggunakan Persamaan (9.5)

$$u_{mo}^{'n} = \begin{cases} 1\ if\ rand \le Sig(\ z_{mo}^{'n}) \\ 0\ \text{otherwise} \end{cases}$$
$$\text{dimana } Sig(\ z_{mo}^{'n}) = 1/(1 + e^{-\ z_{mo}^{'n}}) \qquad (9.5)$$

Langkah 2. Evaluasi posisi setiap partikel yang merupakan kandidat solusi dengan menggunakan fungsi tujuan yang telah ditetapkan (i.e., $fit(u_m^n)$)

Langkah 3. Atur posisi awal dari partikel sebagai posisi personal terbaik (i.e., $pbest_m^n$) berdasarkan hasil dari eveluasi fungsi tujuan, dan pilih posisi terbaik partikel dalam grup/*swarm* untuk disimpan sebagai posisi global terbaik (i.e., $gbest_{mo}^n$)

Langkah 4. Perbaharui kecepatan dari partikel (i.e., z'^n_{mo}) dengan menggunakan fungsi berikut:

$$z'^n_{mo} = \omega z_{mo}^n + \theta_1 rand_1(pbest_{mo}^n - u_{mo}^n) + \theta_2 rand_2(gbest_{mo}^n - u_{mo}^n) \tag{9.6}$$

Langkah 5. Hasilkan populasi partikel yang baru (i.e., *swarm*) berdasarkan Persamaan (9.5).

Langkah 6. Evaluasi kembali posisi setiap partikel yang baru dihasilkan dengan menggunakan fungsi tujuan yang telah ditetapkan (i.e., $fit(u_m^n)$).

Langkah 7. Perbaharui catatan dari setiap partikel dengan membandingkan nilai dari hasil evaluasi (i.e., *fitness value*) dengan posisi personal terbaik (i.e., $pbest_m^n$).

Langkah 8. Perbaharui catatan atas posisi global terbaik (i.e., $gbest_{mo}^n$).

Langkah 9. Ubah *n*=*n*+1. Jika algoritma telah mencapai kriteria berhenti, hentikan iterasi. Jika belum mencapai kriteria, ulangi kembali dari langkah ke 4 hingga mencapai kriteria.

9.2.2. *Modified Binary* PSO (MBPSO)

Shen *et al.* (2004) mengembangkan MBPSO dengan memasukan strategi pembaruan dalam mengubah posisi dari partikel. Strategi pembaruan ini dalam dideskripsikan sebagai berikut:

$$z'^m_{mo} = rand \tag{9.7}$$

$$u'^m_{mo} = \begin{cases} u_{mo}^n if(0 \le z'^m_{mo} \le \tau) \\ pbest_{mo}^n if(\tau < z'^m_{mo} \le \dfrac{1(1+\tau)}{2}) \\ gbest_o^n if(\dfrac{1(1+\tau)}{2} < z'^m_{mo} \le 1) \end{cases} \tag{9.8}$$

τ merupakan bilangan *random* dalam rentang [0,0, 1,0] yang dinamakan sebagai probabilitas statis, dimana nilai awal ditetapkan sebesar 0,5. Skema pembaruan dari perubahan posisi menunjukan mekanisme berbagi dalam partikel. Aturan pertama pada Persamaan (9.8) menjelaskan tentang mekanisme eksplorasi dari MBPSO. Sementara aturan kedua dan ketiga menjaga agar pencarian berbasis eksploitasi lokal tetap terjadi. Probabilitas statis juga digunakan oleh MBPSO untuk menyeimbangkan kemampuan pencarian global dan lokal dari partikel. Nilai yang lebih besar dari τ akan menurunkan probabilitas MBPSO untuk terjebak pada lokal optimal. Di sisi lain, nilai kecil dari τ akan membuat algoritma lebih cepat menuju satu titik (konvergen). Adapun algoritma dari MBPSO dapat dijelaskan sebagai berikut:

Langkah 1. Inisialisasi

- Atur nilai τ, jumlah partikel *N* dalam *swarm* dan jumlah iterasi.
- Tetapkan posisi awal, dimana dalam buku ini ditetapkan sebagai posisi random biner.
- Tetapkan posisi terbaik dan global terbaik sebagai 0.
- Hasilkan posisi awal dari partikel dengan menggunakan Persamaan (9.8)

Langkah 2. Evaluasi posisi setiap partikel yang merupakan kandidat solusi dengan menggunakan fungsi tujuan yang telah ditetapkan (i.e., $fit(u_m^n)$)

Langkah 3. Atur posisi awal dari partikel sebagai posisi personal terbaik (i.e., $pbest_m^n$) berdasarkan hasil dari evaluasi fungsi tujuan, dan pilih posisi terbaik partikel dalam grup/*swarm* untuk disimpan sebagai posisi global terbaik (i.e., $gbest_o^n$)

Langkah 4. Hasilkan populasi partikel yang baru (i.e., *swarm*) berdasarkan Persamaan (9.8)

Langkah 5. Evaluasi kembali setiap partikel yang baru dihasilkan dengan menggunakan funsgi tujuan yang telah ditetapkan (i.e., $fit(u_m^n)$).

Langkah 6. Perbaharui catatan dari setiap partikel dengan membandingkan nilai dari hasil evaluasi (i.e., *fitness value*) dengan posisi personal terbaik (i.e., $pbest_m^n$).

Langkah 7. Perbaharui catatan atas posisi global terbaik (i.e., $gbest_o^n$).

Langkah 8. Ubah *n*=*n*+1. Jika algoritma telah mencapai kriteria berhenti, hentikan iterasi. Jika belum mencapai kriteria, ulangi kembali dari langkah ke 4 hingga mencapai kriteria.

9.2.3. *Probability Discrete Binary* PSO (PBPSO)

Meskipun strategi pembaruan dari MBPSO telah meningkatkan performa PSO dalam hal kecepatan konvergen, MBPSO terkadang terjebak di dalam lokal optimal (Menhas, 2012). Oleh karena itu, Wang *et al.* (2008) mengembangkan PBPSO dengan menggantikan *fungsi sigmoid* pada DBPSO dengan *fungsi transformasi linear*. Fungsi ini digunakan untuk mengestimasi probabilitas aktual dari sebuah *bit* untuk bernilai 1 atau 0. Fungsi dasar dari PBPSO dapat dilihat sebagai berikut:

$$z'^{n}_{mo} = \omega z^{n}_{mo} + \theta_1 rand_1(pbest^{n}_{mo} - u^{n}_{mo}) + \theta_2 rand_2(gbest^{n}_{mo} - u^{n}_{mo}) \tag{9.9}$$

$$z^{n}_{mo} = \begin{cases} vel_{\min} & \text{if } z'^{n}_{mo} \le vel_{\min} \\ z'^{n}_{mo} & \text{if } vel_{\min} < z'^{n}_{mo} < vel_{\max} \\ vel_{\max} & \text{if } vel_{\max} \le z'^{n}_{mo} \end{cases} \tag{9.10}$$

$$w'^{n}_{mo} = w^{n}_{mo} + z^{n}_{mo} \tag{9.11}$$

$$w^{n}_{mo} = \begin{cases} prob_{\min} & \text{if } w'^{n}_{mo} \le prob_{\min} \\ w'^{n}_{mo} & \text{if } prob_{\min} < w'^{n}_{mo} < prob_{\max} \\ prob_{\max} & \text{if } prob_{\max} \le w'^{n}_{mo} \end{cases} \tag{9.12}$$

$$prob_{mo} = (w^{0}_{mo} - prob_{min}())/(probmin_{max}) \tag{9.13}$$

$$u^{1}_{mo} = \begin{cases} 1 if rand \le prob_{mo} (0 \le prob_{mo} \le 1) \\ 0 else \end{cases} \tag{9.14}$$

$prob_{mo}$ merupakan fungsi linear dengan output bernilai di rentang 0 hingga 1. *rand* merupakan nilai stokastik yang diperoleh dari distribusi *uniform* dalam rentang [0,0, 1,0]; [$prob_{max}$,$prob_{min}$] didefinisikan sebagai rentang maksimum dan minimum dari nilai $prob_{mo}$.

Jumlah partikel disimbolkan oleh *pop* dan setiap partikel disimbolkan oleh *m* dimana *m* = 1, 2, . . . ,*pop*. Kemudian $u^{n}_{m} = [u^{n}_{m1}, u^{n}_{m2}, \dots, u^{n}_{mo}, \dots, u^{n}_{mD}]$ merupakan vektor posisi partikel *m* pada saat iterasi ke *n*, dimana *D* merupakan dimensi dari setiap partikel, dan posisi partikel dinotasikan sebagai $u^{n}_{mo} \in \{0,1\}$. Selanjutnya, $w^{n}_{m} = [w^{n}_{m1}, w^{n}_{m2}, \dots, w^{n}_{mo}, \dots, w^{n}_{mD}]$ merupakan posisi desimal dari partikel *m* pada saat iterasi ke *n*. Posisi personal terbaik partikel *m* didefinisikan sebagai $pbest^{n}_{m} = [pbest^{n}_{m1}, pbest^{n}_{m2}, \dots, pbest^{n}_{mo}, \dots, pbest^{n}_{mD}]$ dan $pbest^{n}_{mo} \in \{0,1\}$. Selain itu, $gbest^{n}_{mo}$menunjukkan posisi global terbaik, dimana $gbest^{n}_{o} = [gbest^{n}_{1}, gbest^{nsw}_{2}, \dots, gbest^{n}_{o}, \dots, gbest^{n}_{D}]$ dan $gbest^{n}_{o} \in \{0,1\}$.

$z_m^n = [z_{m1}^n, z_{m2}^n, \ldots, z_{mo}^n, \ldots, z_{mD}^n]$ merupakan vektor kecepatan yang berkaitan dengan partikel *m*. $prob_{max}$ dan $prob_{min}$ merupakan batas bawah dan batas atas dari fungsi probabilitas. Kecepatan maksimum dan minimum didefinisikan sebagai vel_{max} dan vel_{min} untuk membatasi kecepatan. Adapun algoritma dari MBPSO dapat dijelaskan sebagai berikut (Menhas, 2012).

Langkah 1. Initialisasi (*n*=0)

(i) Tentukan nilai dari parameter berikut $\omega, \theta_1, \theta_2, prob_{max}$ dan $prob_{min}$ vel_{max} dan vel_{min}

(ii) Tetapkan vektor kecepatan awal $z_m^0$and vektor awal dari *pseudo* probabilitas initial w_m^0 untuk *m* = 1, 2, . . . , *pop, o*=1, 2, ..., *D* sebagai berikut:

$$z_m^0 = [z_{m1}^0, z_{m2}^0, \ldots, z_{mo}^0, \ldots, z_{mD}^0] \tag{9.15}$$

$$w_m^0 = [w_{m1}^0, w_{m2}^0, \ldots, w_{mo}^0, \ldots, w_{mD}^0] \tag{9.16}$$

$$pbest_m^0 = [pbest_{m1}^0, pbest_{m2}^0, \ldots, pbest_{mo}^0, \ldots, pbest_{mD}^0] \tag{9.17}$$

$$gbest_o^n = [gbest_1^n, gbest_2^n, \ldots, gbest_o^n, \ldots, gbest_D^n] \tag{9.18}$$

Hasilkan *pop* partikel yang merupakan vektor kandidat solusi $\{u_m^1 = [u_{m1}^1, u_{m2}^1, \ldots, u_{mo}^1, \ldots, u_{mD}^1]\}$ menggunakan persamaan berikut:

$$prob_{mo} = (w_{mo}^0 - prob_{min})/(prob_{max}) \tag{9.19}$$

$$u_{mo}^1 = \begin{cases} 1 if\ rand \le prob_{mo} (0 \le prob_{mo} \le 1) \\ 0\ else \end{cases} \tag{9.20}$$

(iii) Set $z_m^0 = z_m^1$ dan $w_m^0 = w_m^1$.

Langkah 2. Evaluasi posisi setiap partikel yang merupakan kandidat solusi dengan menggunakan fungsi tujuan yang telah ditetapkan (i.e., $fit(u_m^n)$)

Langkah 3. Atur posisi awal dari partikel sebagai posisi personal terbaik (i.e., $pbest_m^n$) berdasarkan hasil dari evaluasi fungsi tujuan, dan pilih posisi terbaik partikel dalam grup/*swarm* untuk disimpan sebagai posisi global terbaik (i.e., $gbest_{mo}^n$)

Langkah 4. Perbaharui kecepatan dari partikel (i.e., z'^n_{mo}) dengan menggunakan persamaan berikut:

$$z'^n_{mo} = \omega z_{mo}^n + \theta_1 rand_1 (pbest_{mo}^n - u_{mo}^n) + \theta_2 rand_2 (gbest_{mo}^n - u_{mo}^n) \tag{9.21}$$

$$z_{mo}^n = \begin{cases} vel_{\min} & \text{if } z'^n_{mo} \le vel_{\min} \\ z'^n_{mo} & \text{if } vel_{\min} < z'^n_{mo} < vel_{\max} \\ vel_{\max} & \text{if } vel_{\max} \le z'^n_{mo} \end{cases} \tag{9.22}$$

$$w''^{n}_{mo} = w^{n}_{mo} + z^{n}_{mo} \tag{9.23}$$

$$w^{n}_{mo} = \begin{cases} prob_{\min} & \text{if } w'^{n}_{mo} \le prob_{\min} \\ w'^{n}_{mo} & \text{if } prob_{\min} < w'^{n}_{mo} < prob_{\max} \\ prob_{\max} & \text{if } prob_{\max} \le w'^{n}_{mo} \end{cases} \tag{9.24}$$

Langkah 5. Hasilkan populasi partikel yang baru (i.e., *swarm*) berdasarkan Persamaan (9.19) dan (9.20) kemudian evaluasi kembali$fit\left(u''^{n}_{m}\right)$.

Langkah 8. Perbaharui catatan dari setiap partikel dengan membandingkan nilai dari hasil evaluasi (i.e., *fitness value*) dengan posisi personal terbaik (i.e., $pbest^{n}_{m}$), Perbaharui catatan atas posisi global terbaik (i.e., $gbest^{n}_{mo}$).

Langkah 9. Ubah *n*=*n*+1. Jika algoritma telah mencapai kriteria berhenti, hentikan iterasi. Jika belum mencapai kriteria, ulangi kembali dari langkah ke 4 hingga mencapai kriteria.

9.2.1 *Modified* PBPSO (MPBPSO)

MPBPSO (Zukhruf, 2014) merupakan algoritma yang memasukkan strategi pembaruan yang terinspirasi dari MBPSO kedalam PBPSO. Sehingga metode ini diharapkan memiliki karakteristik yang baik dalam hal kecepatan konvergensi diturunkan dari skema pembaruan MBPSO, dan dalam hal kemampuan mencari solusi diturunkan dari karakter PBPSO. Strategi pembaruan dari MBPSO dapat dilihat dalam persamaan berikut:

$$(0 \le rand < \tau) \text{ then } u''^{n}_{mo} = gbest^{n}_{o} \tag{9.25}$$

$$if\ (\tau \le rand < (2\tau + 1)/3) \text{ then } u''^{n}_{mo} = pbest^{n}_{mo} \tag{9.26}$$

$$if\ ((2\tau + 1)/3 \le rand < (\tau + 2)/3) \text{ then } u''^{n}_{mo} = irand \tag{9.27}$$

$$if\ (\tau + 2)/3 \le rand \le 1 \text{ then } u''^{n}_{mo} = u^{n}_{mo} \tag{9.28}$$

$rand$ merupakan bilangan *random* biner (0 atau 1). Persamaan (9.25)–(9.26) menjaga arah dari setiap partikel untuk mengikuti solusi terbaik dan mendorong partikel untuk mencapai titik konvergen lebih cepat. Keseimbangan dalam kemampuan pencarian global dan pencarian lokal sangat mempengaruhi performa PSO. Oleh karena itu, Persamaan (9.27) digunakan untuk memperluas ruang pencarian. Shen *et al.* (2004, 2009) mengindikasikan pentingnya untuk meningkatkan kemampuan pencarian global dengan memaksa 10% dari partikel terbang bebas mencari solusi. Oleh karena itu, MPBPSO menambahkan bilangan *random* biner $rand$ untuk meningkatkan kemampuan pencarian,

yang tidak dimiliki oleh MBPSO. Beberapa elemen dari partikel pun dipaksa untuk tetap berdasarkan Persamaan (9.28). Tanpa kedua proses ini, partikel akan cenderung untuk mengikuti posisi terbaik mereka yang lalu dan berpotensi terjebak dalam lokal optimal. Adapun prosedur penggunaan MPBPSO dapat dilihat sebagai berikut:

Langkah 1. Initialisasi (n=0)

(i) Tentukan nilai dari parameter berikut $\omega, \theta_1, \theta_2, prob_{max}$ dan $prob_{min}$, vel_{max} dan vel_{min}.

(ii) Tetapkan vektor kecepatan awal z_m^0 dan vektor awal dari probabilitas semu initial w_m^0 untuk m = 1, 2, . . . , *pop*, o=1, 2, ..., D sebagai berikut:

$$z_m^0 = [z_{m1}^0, z_{m2}^0, \ldots, z_{mo}^0, \ldots, z_{mD}^0] \tag{9.29}$$

$$w_m^0 = [w_{m1}^0, w_{m2}^0, \ldots, w_{mo}^0, \ldots, w_{mD}^0] \tag{9.30}$$

$$pbest_m^0 = [pbest_{m1}^0, pbest_{m2}^0, \ldots, pbest_{mo}^0, \ldots, pbest_{mD}^0] \tag{9.31}$$

$$gbest_o^n = [gbest_1^n, gbest_2^n, \ldots, gbest_o^n, \ldots, gbest_D^n] \tag{9.32}$$

(iii) Hasilkan *pop* partikel yang merupakan vektor kandidat solusi $u_m^1 = [u_{m1}^1, u_{m2}^1, \ldots, u_{mo}^1, \ldots, u_{mD}^1]$ menggunakan persamaan berikut:

$$prob_{mo} = (w_{mo}^0 - prob_{min})/(probmin_{max}) \tag{9.33}$$

$$u_{mo}^1 = \begin{cases} 1 if\ rand \le prob_{mo} (0 \le prob_{mo} \le 1) \\ 0\ else \end{cases} \tag{9.34}$$

(iv) $Set\ z_m^0 = z_m^1$ dan $w_m^0 = w_m^1$.

Langkah 2. Evaluasi posisi setiap partikel yang merupakan kandidat solusi dengan menggunakan fungsi tujuan yang telah ditetapkan (i.e., $fit(u_m^n)$)

Langkah 3. Atur posisi awal dari partikel sebagai posisi personal terbaik (i.e., $pbest_m^n$) berdasarkan hasil dari eveluasi fungsi tujuan, dan pilih posisi terbaik partikel dalam grup/*swarm* untuk disimpan sebagai posisi global terbaik (i.e., $gbest_o^n$)

Langkah 4. Perbaharui kecepatan dari partikel (i.e., z'^n_{mo}) dengan menggunakan Persamaan berikut:

$$z'^n_{mo} = \omega z_{mo}^n + \theta_1 rand_1 (pbest_{mo}^n - u_{mo}^n) + \theta_2 rand_2 (gbest_{mo}^n - u_{mo}^n) \tag{9.35}$$

$$z_{mo}^n = \begin{cases} vel_{\min} & \text{if } z'^n_{mo} \le vel_{\min} \\ z'^n_{mo} & \text{if } vel_{\min} < z'^n_{mo} < vel_{\max} \\ vel_{\max} & \text{if } vel_{\max} \le z'^n_{mo} \end{cases} \tag{9.36}$$

$$w'^n_{mo} = w_{mo}^n + z_{mo}^n \tag{9.37}$$

$$w_{mo}^{n} = \begin{cases} prob_{\min} \text{ if } w'^{n}_{mo} \le prob_{\min} \\ w'^{n}_{mo} \quad \text{if } prob_{\min} < w'^{n}_{mo} < prob_{\max} \\ prob_{\max} \text{ if } prob_{\max} \le w'^{n}_{mo} \end{cases} \tag{9.38}$$

Langkah 5. Hasilkan populasi partikel yang baru (i.e., *swarm*) berdasarkan langkah (9.34) kemudian evaluasi kembali$fit\left(u''^{n}_{m}\right)$.

Langkah 6. Jika $n \ge 1$, cek kembali nilai evaluasi dari setiap partikel. Jika $fit\left(u'^{n}_{m}\right) - fit(u^{n}_{m}) \le 0$, perbarui posisi partikel u''^{n}_{m} menggunakan aturan berikut:

Ulangi sebanyak *D* kali (*D*=panjang *bit* partikel)

$$u''^{n}_{mo} = \begin{cases} gbest^{n}_{o} \; if \; 0 \le rand < \tau \\ pbest^{n}_{mo} if \; \tau \le rand < (2\tau + 1)/3 \\ rand \; if \; (2\tau + 1)/3 \le rand < (\tau + 2)/3 \\ u^{n}_{mo} \; if \; (\tau + 2)/3 \le rand \le 1 \end{cases} \tag{9.39}$$

dimana $\tau = \tau_1 + \tau_2 \times rand$

Langkah 7. Evaluasi $fit\left(u'^{n}_{m}\right)$. Kemudia, update u^{n}_{m} sebagai berikut:

$$u^{n}_{m} = \begin{cases} u^{n}_{m} \; if \; fit\left(u''^{n}_{m}\right) < fit(u^{n}_{m}) \\ u''^{n}_{m} \; if \; fit\left(u''^{n}_{m}\right) \ge fit(u^{n}_{m}) \end{cases} \tag{9.40}$$

Langkah 8. Perbaharui catatan dari setiap partikel dengan membandingkan nilai dari hasil evaluasi (i.e., *fitness value*) dengan posisi personal terbaik (i.e., $pbest^{n}_{m}$), Perbaharui catatan atas posisi global terbaik (i.e., $gbest^{n}_{o}$).

Langkah 9. Ubah *n*=*n*+1. Jika algoritma telah mencapai kriteria berhenti, hentikan iterasi. Jika belum mencapai kriteria, ulangi kembali dari langkah ke 4 hingga mencapai kriteria.

9.3. Aplikasi PSO dalam *Knapsack Problem*

9.3.1. Contoh *Knapsack*

Pada bagian ini, algoritma PSO akan diaplikasikan untuk memecahkan permasalahan *Knapsack*. Seperti yang telah disebutkan di atas, *Knapsack* merupakan masalah optimasi yang bertujuan untuk membawa barang dengan berat tertentu pada sebuah/beberapa ransel agar total nilai dari barang yang dibawa tersebut maksimal. Fungsi tujuan dari permasalahan ini dapat dituliskan sebagai berikut:

$$\underset{u}{Max} \quad \sum_{i=1}^{I} h_i \, x_i \tag{9.37}$$

$$\text{subject to } \sum_{i=1}^{I} b_i \, x_i \leq B, x_i \in \{0,1\} \tag{9.38}$$

Dimana h_i menunjukan nilai dari setiap barang i yang memiliki berat b_i. B menggambarkan berat/kapasitas maksimum yang dimiliki oleh ransel. Sementara x_i menunjukkan apakah barang i dimasukkan kedalam ransel atau tidak, dimana $x_i = 1$ berarti barang *i* dimasukkan ke dalam ransel dan sebaliknya.

Tabel 9.1 memperlihatkan contoh dari permasalahan *Knapsack* , dimana terdapat 16 barang dengan berat dan nilai tertentu. Pembawa ransel ingin memasukkan barang ke dalam ransel berkapasistas 80 dengan memaksimalkan nilai dari kombinasi barang tersebut. Asumsikan bahwa berat dalam satuan Kg dan nilai barang bersatuan 100,000 rupiah. Karena permasalahan ini memberikan batasan maksimum berat yang dapat dipikul oleh ransel, maka fungsi penalti perlu ditambahkan dalam algoritma PSO. Sesuai namanya, fungsi penalti berperan memberikan penalti kepada partikel yang jumlah beratnya melebih kapasitas maksimum ransel. Terdapat bermacam-macam bentuk dari fungsi penalti (lihat Olsen, 1994), meskipun pada buku ini digunakan fungsi penalti sederhana seperti yang terlihat pada Persamaan (9.39).

$$\sum_{i=1}^{I} h_i \, x_i \; = \sum_{i=1}^{I} h_i \, x_i - (\alpha(\sum_{i=1}^{I} b_i \, x_i - B)) \tag{9.39}$$

Nilai α harus ditentukan melalui proses iterasi untuk menemukan nilai alpha yang sesuai.

Tabel 9.1 Contoh permasalahan *Knapsack*.

No (i)	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Berat (b_i)	11	2	13	15	11	12	12	7	11	4	11	2	5	2	3	13
Nilai (h_i)	33	31	43	44	25	35	33	40	41	43	28	41	40	25	23	35

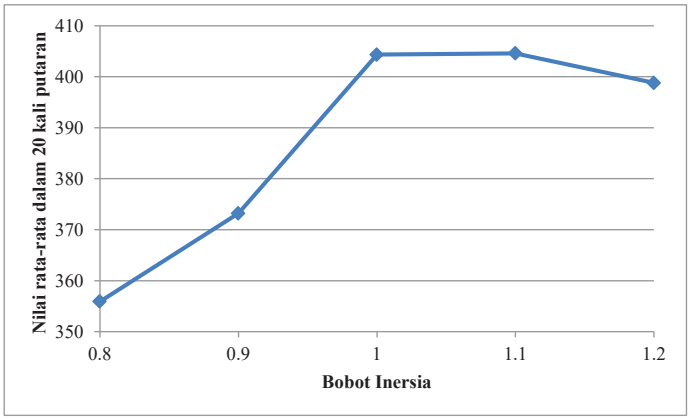
Kapasitas ransel: B=80

9.3.2. Analisis Hasil dan Pengaturan Parameter PSO

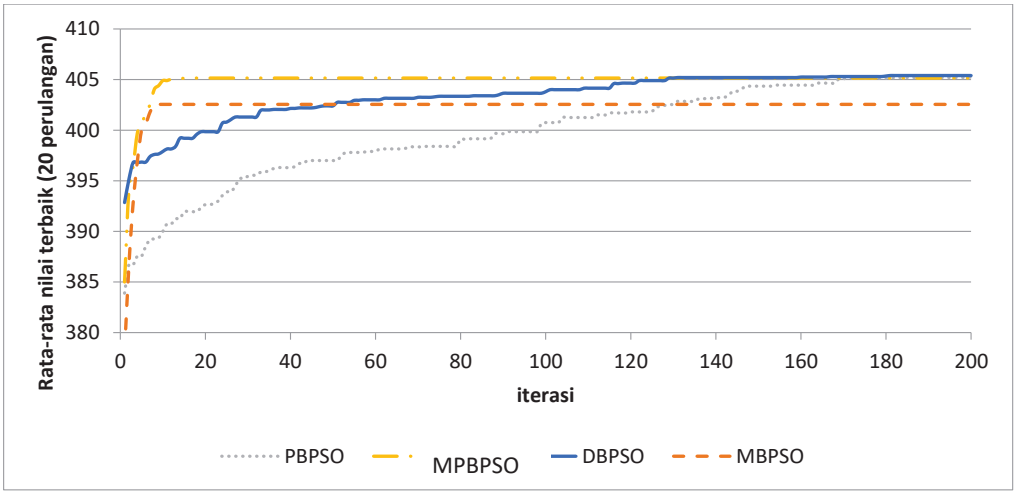
Bagian ini akan membahas hasil dari model optimasi barang bawaan dalam permasalahan *knapsack* dengan menggunakan algoritma PSO. Untuk memastikan hasil yang eksak, dilakukan pula proses enumerasi sempurna. Proses enumerasi sempurna ini mencoba 65.535 kombinasi yang mungkin. Solusi eksak dari hasil enumerasi sempurna menemukan

bahwa membawa barang no 1, 2, 3, 6, 8, 9, 10, 11, 12, 13, 14, 15 akan memberikan nilai ransel maksimum sebesar Rp. 406 (dalam Rp. 100.000). Setiap algoritma PSO (i.e., DBPSO, MBPSO, PBPSO, MPBPSO) dapat memperoleh solusi seperti yang dihasilkan oleh metode enumerasi sempurna dengan waktu perhitungan yang jauh lebih kecil daripada metode enumerasi sempurana.

Meskipun akan terselip pertanyaan, bagaimana menentukan setiap parameter dari PSO agar dapat memberikan performa terbaik? Dan algoritma PSO mana yang mampu memberikan performa yang lebih baik? Karena performa PSO sangat tergantung dari parameter yang ditentukan, pengguna metode ini wajib untuk melakukan tes sensitifitas, atau *numerical test* untuk menemukan parameter terbaik. Karena PSO mengandung proses acak yang mempengaruhi kestabilannya, *numerical test* ini dapat dilakukan dengan mengulang-ulang perhitungan sebanyak 10-20 kali untuk mengetahui kestabilan algoritma PSO. Test ini dilakukan dengan mengubah-ubah nilai setiap parameter dari PSO untuk menemukan parameter yang sesuai dan memberikan hasil terbaik. Gambar 9.2 memberikan contoh proses penentuan parameter terbaik pada MPBPSO. Pada gambar tersebut dengan mengasumsikan bahwa parameter lain tetap dan nilai bobot inersia berubah, diperoleh bahwa nilai bobot inersia 1.0−1.1 dapat memberikan hasil yang lebih stabil dibandingkan nilai lainnya.



Gambar 9.2 Contoh pengaturan parameter MPBPSO.



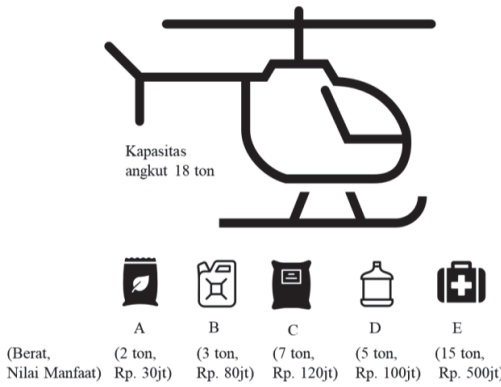
Gambar 9.3 Perbandingan performa dari algortima PSO.

Setelah melakukan tes pada setiap parameter pada setiap algoritma, performa dari setiap algoritma sesungguhnya dimungkinkan untuk dibandingkan. Pada Gambar 9.3 dapat dilihat bahwa MPBPSO dapat mencapai nilai terbaik pada iterasi/putaran yang lebih kecil, sehingga dapat memperkecil waktu perhitungan. Meskipun tes selanjutnya penting untuk dilakukan dalam rangka membandingkan performa algortima PSO.

9.4. Aplikasi PSO dengan *Ms. Excel*

Pada bagian ini, PSO akan diterapkan untuk menyelesaikan permasalahan pengangkutan bahan bantuan untuk korban bencana alam dengan helikopter. Anggaplah terdapat 5 jenis barang dalam bentuk paket-paket serta dengan berat yang berbeda-beda (W), dengan nilai manfaat dikuantifikasikan dalam bentuk rupiah (V). Meskipun helikopter tersebut memiliki batasan dari sisi kapasitas angkut (lihat Gambar 9.4). Sehingga menjadi penting untuk menentukan barang mana saja yang dapat dibawa dalam rangka memaksimalkan nilai harganya. Oleh karenanya permasalahan optimasi ini berusaha menentukan barang apa saja yang perlu dimasukkan ke dalam helikopter dalam rangka memaksimalkan nilai harga yang dapat dibawa helikopter tersebut.

Permasalahan ini sesungguhnya berjenis serupa dengan *binary knapsack*. Permasalahan *binary knapsack* pada prinsipnya memiliki struktur yang kesesuaian dengan beberapa permasalahan optimasi di bidang transportasi.



Gambar 9.4 Contoh permasalahan optimasi pengakutan barang dengan kapasitas angkut terbatas.

Permasalahan optimasi pada contoh ini termasuk permasalahan dengan adanya fungsi kendala. Sehingga pada PSO akan diterapkan penalti dalam rangka menjaga bahwa barang-barang yang dimasukkan tidak melebihi kapasitas yang telah ditentukan. Penentuan fungsi penalti ini pada prinsipnya memerlukan elaborasi yang mendalam. Meskipun dalam bagian ini akan dibuat fungsi penalti sederhana pada fungsi objektif dengan cara mengalikan *fitness value* dengan 0,05. Tahapan pertama untuk menyelesaikan permasalahan ini adalah mendefinisikan menjadi 3 fitur utama, yaitu *objective function, decision variable,* dan *constraints*.

- *Objective function:* memaksimalkan nilai harga yang dapat dibawa oleh helikopter
- *Decision variable:* barang mana yang akan dimasukkan kedalam helikopter
- *Constraint:* kapasitas helikopter 18 ton.

Untuk menyelesaikan permasalahan dengan menggunakan *Ms. Excel* dapat dilakukan beberapa tahapan berikut:

- **Langkah 1:** Tetapkan bentuk *decision variable* dalam *Ms. Excel*

  Pada permasalahan ini direpresentasikan dengan pernyataan YES/NO. YES berarti paket barang dimasukkan kedalam helikopter, sementara NO jika tidak dimasukkan. Untuk memudahkan proses perhitungan pernyataan YES/NO diubah menjadi 1/0. Untuk pernyataan YES diasumsikan nilai 1 sedangkan NO diasumsikan nilai 0. Nilai 0/1 ini kemudian mendefinisikan informasi *bit* pada setiap posisi dari setiap partikel (lihat Tabel 9.2). Sehingga setiap partikel akan memiliki panjang *bit* sebesar 5 yang merepresentasikan keputusan yang harus diambil oleh partikel.

Tabel 9.2 Representasi data *knapsack*.

A	B	C	D	E
Yes/No	Yes/No	Yes/No	Yes/No	Yes/No
1/0	1/0	1/0	1/0	1/0

- **Langkah 2:** Formulasikan *Objective function* dan *Constraint* dalam *equation* pada Ms. Excel

  Pada permasalahan ini, *objective function*-nya yaitu paket barang apa saja yang dapat masuk untuk memaksimalkan nilai manfaat dari bawaan tersebut. Rumusnya yaitu:

$$MAX\ ((V1 * bit\ 1) + (V2\ * bit\ 2) + (V3\ * bit\ 3) + (V4 * bit\ 4) + (V5 * bit\ 5)). \tag{9.40}$$

  Dimana *V1*,*V2*,*V3*,*V4*, dan *V5* menunjukkan nilai dari barang yang akan dimasukkan. Sedangkan untuk *constraints*-nya yaitu terdapat kendala bahwa helikopter hanya bisa menampung 18 ton sehingga barang apa saja yang bisa masuk dapat diformulasikan sebagai berikut. Dimana *W1*, *W2*, *W3*, *W4*, dan *W5* menunjukkan berat dari masing-masing barang yang akan dimasukkan

$$18ton\ \geq\ (W1 * bit\ 1) + (W2 * bit\ 2) + (W3 * bit\ 3) + (W4\ * bit\ 4) + (W5\ * bit\ 5)) \tag{9.41}$$

- **Langkah 3:** *Setting parameter* DBPSO

  Pada buku ini dijelaskan proses perhitungan dengan menggunakan DBPSO, dimana tahap pertama yang harus dilakukan adalah menentukan parameter untuk proses iterasi DBPSO. Penentuan parameter ini pada prinsipnya harus didasarkan kepada percobaan untuk menentukan nilai parameter yang terbaik (lihat bagian sebelumnya). Meskipun dalam contoh ini, parameter dianggap telah ditentukan sebagai berikut:

Tabel 9.3 Parameter settings PSO.

Length of Particle	5
Number of Particle	5
Number of Iteration	6
Learning Factor (θ)	2
Maximum Velocity	6
Minimum Velocity	-6
Inertia Weight (*w*)	1

- **Langkah 4:** Inisialisasi parameter kecepatan partikel

  Parameter kecepatan berperan besar dalam menentukan posisi dari masing-masing partikel yang merepresentasikan kandidat solusi masalah optimasi. Pada DBPSO nilai awal kecepatan umumnya ditetapkan nol, yang akan membuat setiap *bit* dalam partikel memiliki peluang 50% untuk bernilai 1 ataupun bernilai 0.

Tabel 9.4 Tabel untuk *velocity* setiap partikel.

	A	B	C	D	E
Velocity Partikel 1	0	0	0	0	0
Velocity Partikel 2	0	0	0	0	0
Velocity Partikel 3	0	0	0	0	0
Velocity Partikel 4	0	0	0	0	0
Velocity Partikel 5	0	0	0	0	0

- **Langkah 5:** Perbaharui nilai dari fungsi sigmoid pada masing-masing *bit*

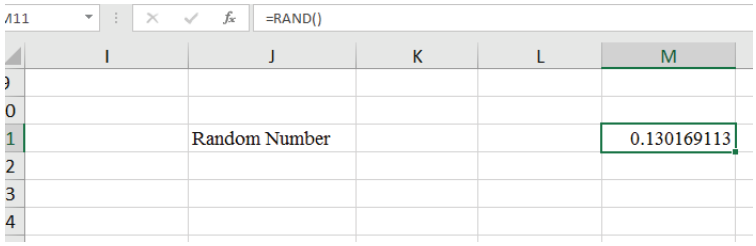
  Posisi dari setiap partikel diperbaharui dengan menggunakan fungsi sigmoid (Persamaan 9.2).

Tabel 9.5 Tabel untuk fungsi sigmoid pada kondisi awal.

	A	B	C	D	E
Nilai sigmoid partikel 1	0.5	0.5	0.5	0.5	0.5
Nilai sigmoid partikel 2	0.5	0.5	0.5	0.5	0.5
Nilai sigmoid partikel 3	0.5	0.5	0.5	0.5	0.5
Nilai sigmoid partikel 4	0.5	0.5	0.5	0.5	0.5
Nilai sigmoid partikel 5	0.5	0.5	0.5	0.5	0.5

- **Langkah 6:** Perbaharui posisi biner pada setiap *bit* pada setiap partikel

  Setelah mendapatkan nilai dari *sigmoid*, posisi dari partikel dapat diperbaharui dengan menggunakan bilangan acak dan mengkonsiderasikan Persamaan (9.3). Secara teknis proses penggunaan bilangan acak pada *Ms. Excel* dapat dilakukan dengan menggunakan fungsi *rand ()* seperti yang diilustrasikan pada gambar dibawah ini. Meskipun dapat pula menggunakan bilangan acak yang dirumuskan sendiri.



Gambar 9.5 Rumus pada *Ms. Excel* untuk membangkitkan bilangan acak.

Fungsi ini kemudian dapat digunakan untuk membangkitkan bilangan acak pada setiap *bit* pada setiap partikel. Secara teknis hal tersebut dilakukan dengan *copy+paste* kepada tabel yang memuat rumus bilangan acak ke setiap *bit* dan setiap partikel (lihat Tabel 9.6). Untuk mencegah bilangan acak ini berubah-ubah, gunakan *paste value*, setelah meng-*copy* fungsi tersebut.

Tabel 9.6 Tabel untuk bilangan acak.

	A	B	C	D	E
Bilangan acak partikel 1	0.52	0.78	0.81	0.49	0.38
Bilangan acak partikel 2	0.84	0.63	0.69	0.41	0.53
Bilangan acak partikel 3	0.96	0.56	0.06	0.10	0.44
Bilangan acak partikel 4	0.75	0.31	0.92	0.82	0.12
Bilangan acak partikel 5	0.42	0.26	0.07	0.10	0.66

Bilangan acak pada setiap *bit* pada setiap partikel kemudian digunakan untuk memperbaharui posisi partikel dengan menerapkan Persamaan (9.3). Persamaan tersebut menyebutkan jika nilai bilangan acak kurang dari nilai sigmoid maka posisi *bit* bernilai 1 dan sebaliknya, hasil dari proses pembaharuan ini terlihat pada Tabel 9.7.

Tabel 9.7 Tabel *posisi partikel.*

	A	B	C	D	E
Posisi Partikel 1	0	0	0	1	1
Posisi Partikel 2	0	0	0	1	0
Posisi Partikel 3	0	0	1	1	1
Posisi Partikel 4	0	1	0	0	1
Posisi Partikel 5	1	1	1	1	0

- **Langkah 7:** Evaluasi *fitness value* dari posisi partikel

  Posisi partikel pada prinsipnya adalah kandidat solusi sehingga penting untuk dievaluasi seberapa baik kandidat solusi tersebut. Evaluasi umumnya dilakukan dengan memasukkan *decision variable* ke dalam *objective function* yang dikenal sebagai *fitness value*. Perhitungan *fitness value* dapat menggunakan Persamaan (9.40), sementara untuk mengevaluasi apakah kandidat solusi melebihi kendala kapasitas yang ditetapkan gunakan Persamaan (9.41). Jika kandidat solusi memiliki berat yang lebih besar daripada batasan kapasitas maka *fitness value* akan di-*penalty* dengan mengalikan *fitness value*-nya dengan 0,05. Resume dari penerapannya dapat dilihat pada Tabel 9.8.

Tabel 9.8 Tabel *fitness value* dan *evaluasi fungsi kendala.*

Particle	***Fitness value* tanpa penalti**	***constraint check***	**Cek *constraint***	***Fitness value***
Partikel 1	600	20	*Penalty*	30
Partikel 2	100	5	*OK*	100
Partikel 3	720	27	*Penalty*	36
Partikel 4	580	18	*OK*	580
Partikel 5	320	17	*OK*	320

- **Langkah 8:** Perbaharui *Pbest*

  Langkah selanjutnya adalah melakukan pembaharuan pada *Pbest* yang merepresentasikan posisi terbaik yang diperoleh oleh masing-masing partikel selama proses iterasi. Pada kondisi iterasi pertama nilai *Pbest* sama dengan posisi saat ini.

Meskipun pada iterasi selanjutnya, pembaharuan *Pbest* dilakukan dengan mengecek apakah *fitness value* dari posisi saat ini lebih baik dibandingkan dengan *Pbest* pada iterasi sebelumnya. Jika lebih baik maka perbaharui *Pbest* jika tidak maka *Pbest* tidak perlu diperbaharui.

Tabel 9.9 Tabel *Pbest*.

	A	B	C	D	E
Pbest Partikel 1	0	0	0	1	1
Pbest Partikel 2	0	0	0	1	0
Pbest Partikel 3	0	0	1	1	1
Pbest Partikel 4	0	1	0	0	1
Pbest Partikel 5	1	1	1	1	0

- **Langkah 9:** Perbaharui *Gbest*

  Setelah nilai *Pbest* diperbaharui, maka langkah selanjutnya adalah memperbaharui nilai *Gbest* yang merupakan posisi terbaik yang pernah diperoleh oleh seluruh partikel selama iterasi. Posisi ini diperoleh dengan membandingkan apakah *Gbest* pada iterasi sebelumnya masih lebih besar dibandingkan nilai *Pbest* masing-masing partikel. Jika lebih besar maka perbaharui posisi *Gbest*, jika tidak maka *Gbest* iterasi saat ini sama seperti *Gbest* sebelumnya. Jika pada iterasi pertama, maka nilai *Gbest* sama dengan nilai *Pbest* yang memiliki *fitness value* terbesar.

Update Gbest

Gbest Iter 1	0	1	0	0	1

fitness value	constrain check		fitness value
580	18	OK	580

Gambar 9.6 Nilai terbaik dari pergerakan pertama di iterasi 1.

- **Langkah 10:** Perbaharui *Velocity*

  Setelah *Gbest* diperoleh maka posisi partikel untuk iterasi selanjutnya dapat diperbaharui dengan memperbaharui terlebih dahulu kecepatan masing-masing partikel menggunakan Persamaan (9.1). Persamaan tersebut membutuhkan informasi *Pbest* (Tabel 9.9), *Gbest* (Gambar 9.6), posisi sebelumnya (Tabel 9.7), *inertia weight, learning factor* (Tabel 9.3) dan bilangan acak yang dibangkitkan untuk setiap *bit* setiap partikel. Adapun resume dari *velocity* dapat dilihat pada Tabel 9.10.

Tabel 9.10 Tabel pembaharuan kecepatan.

Kecepatan Partikel 1	0	1.4088171	0	-1.224994	0
Kecepatan Partikel 12	0	1.9775404	0	-0.755899	0.0737148
Kecepatan Partikel 13	0	0.7663925	-0.779821	-1.67609	0
Kecepatan Partikel 14	0	0	0	0	0
Kecepatan Partikel 15	-0.272667	0	-0.859057	-1.099449	0.041357

- **Langkah 11:** Perbaharui iterasi, jika nilai iterasi>total iterasi yang ditentukan, proses perhitungan selesai dan gunakan *Gbest* sebagai solusi. Jika tidak, kembali ke langkah 5.

Template penggunaan PSO dengan *Ms. Excel* dapat diperoleh pada laman berikut:

https://*bit*.ly/psotempl

Pembaca yang tertarik untuk melihat penjelasan dalam bentuk audio-visual terkait Bab ini dapat mengunjungi playlist youtube berikut:

https://bit.ly/Playlist-PSO

10. *Genetic Algorithm* (GA)

10.1. Pendahuluan

Genetic Algorithm (GA) adalah salah satu metode yang berbasis *metaheuristik* yang dikembangkan oleh oleh Holland et al (1975) untuk melakukan proses optimasi. GA terinspirasi oleh seleksi alam dan proses genetika, secara spesifik berupa peristiwa seleksi, persilangan dan mutasi. Selain itu, adanya informasi biologis yang diturunkan secara turun-temurun dari satu generasi ke generasi turut menjadi dasar inspirasi GA. Pemahaman GA secara umum dapat diresumekan ke dalam 4 tahap yang cukup berkaitan erat dengan teori evolusi Darwin.

1. *Reproduction:* Adanya individu yang berkembang biak sehingga menghasilkan keturunan.
2. *Competition:* Adanya keturunan dari individu yang berkompetisi satu sama lain.
3. *Selection:* Adanya kompetisi secara alami menyebabkan setiap individu akan mengalami proses seleksi alam agar tetap bertahan hidup.
4. *Survive:* Individu yang bertahan akan melanjutkan kepada tahapan reproduksi untuk menciptakan keturunan baru.

Peristiwa alam ini kemudian berusaha diimplementasikan ke dalam sebuah algoritma yang mampu melakukan pencarian solusi dalam permasalahan optimasi. Secara umum, permasalahan optimasi yang dapat ditangani oleh GA berada dari bentuk *continuous optimization* hingga *discrete optimization.* Pada bidang transportasi, penggunaan GA biasanya bertipe *discrete optimization* yang dibahas di dalam buku ini. Secara spesifik buku ini akan membahas *binary discrete optimization*.

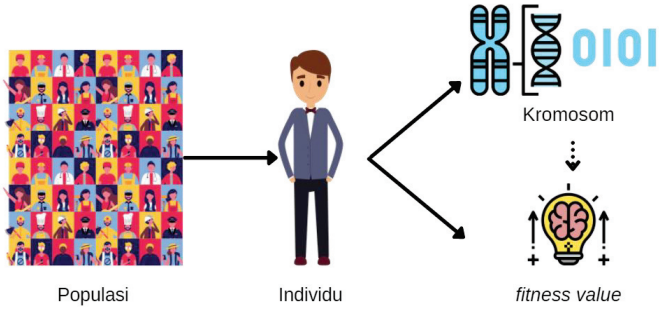
10.2. Kerangka dasar Genetic Algorithm

Sebelum menelisik lebih dalam kepada proses implementasi GA, terdapat beberapa istilah yang penting untuk dipahami.

- *Populasi*: kumpulan dari beberapa individu
- *Individu*: kandidat solusi yang memiliki dua informasi dasar yaitu *fitness value* dan *chromosome*.
- *Chromosome*: informasi genetik yang ada pada suatu individu, ditandai oleh nilai 0 atau 1 untuk permasalahan optimasi biner.

- *Gen*: *chromosome* apabila dibagi-bagi menjadi bagian yang lebih kecil maka akan disebut *gen*.
- *Allele:* bagian lebih kecil dari *gen* atau ada pula yang mendefinisikan sebagai nilai dari *gen* (1 atau 0)
- *fitness value:* nilai fungsi objektif jika dimasukkan kandidat solusi yang diberikan oleh *chromosome.* Sehingga *fitness value* juga melambangkan nilai dari suatu individu terhadap permasalahan optimasi.

Gambar 10.1 kemudian memberikan ilustrasi secara umum terkait fitur penyusun populasi dalam GA. Sehingga secara umum populasi dibentuk oleh individu, dimana individu tersebut memiliki dua informasi yaitu *chromosome* dan *fitness value.* Kemudian nilai *fitness* akan sangat ditentukan oleh informasi yang termaktub di dalam *chromosome* suatu individu. (lihat Gambar 10.1).

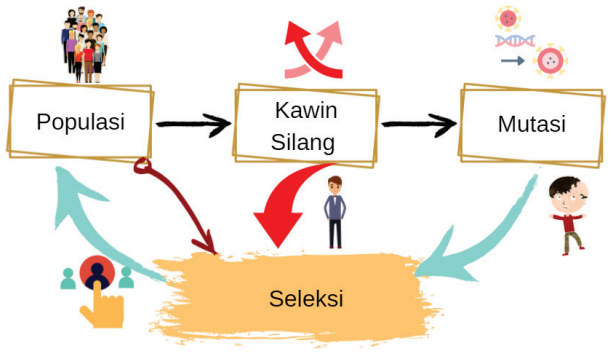


Gambar 10.1 Ilustrasi terkait istilah penyusun *Genetic Algorithm.*

Proses dalam GA berkorelasi dengan teori seleksi alam, dimana adanya individu yang menyusun populasi, kemudian terjadinya perkawinan antar individu. Kemudian tahap perkawinan ini ditandai dengan adanya proses persilangan (*crossover*) untuk menukarkan informasi genetika antar individu. Keturunan hasil persilangan dimungkinkan terjadinya mutasi atau perubahan genetika. Semua individu dalam populasi kemudian saling berkompetisi satu sama lain agar dapat bertahan dalam kerangka proses seleksi. Proses ini diharapkan pada akhir generasi keturunan yang bertahan merupakan individu terbaik dengan informasi genetik terbaik. Atas dasar proses ini, GA umumnya memiliki tiga operator dasar yaitu:

- *Selection;*
- *Crossover;*
- *Mutation.*

Meskipun dalam perkembangannya GA memiliki operator yang lebih dari tiga, akan tetapi ketiga operator ini merupakan operator dasar yang menyusun GA. Interaksi ketiga operator tersebut dalam GA dapat diilustrasikan pada Gambar 10.2



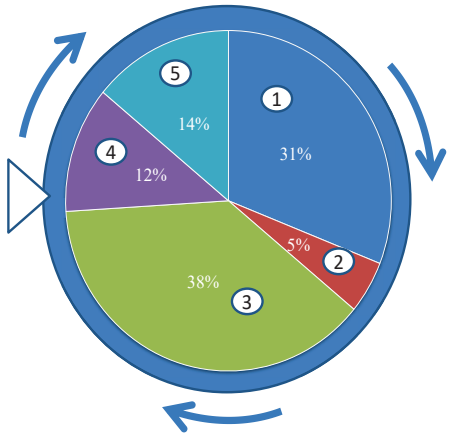
Gambar 10.2 Interaksi tiga operator utama dalam GA.

9.2.1. *Operator Selection*

Operator *Selection* merupakan salah satu operator dasar dari GA. *Selection* berperan menyeleksi individu mana yang bertahan kemudian melanjutkan ke generasi selanjutnya. Untuk melakukan proses seleksi, GA selalu melibatkan proses acak dalam rangka memilih individu yang akan bertahan pada generasi selanjutnya. Terdapat berbagai cara untuk memilih individu secara acak, meskipun penggunaan roda rolet yang diputar umum digunakan. Dalam konteks seleksi, individu yang akan melanjutkan ke generasi selanjutnya dipilih secara acak dengan roda rolet. Proses seleksi menjadi salah satu bagian yang penting di dalam GA, karena di dalam GA jumlah individu harus tetap dalam sebuah populasi hingga akhir generasi. Sebagai contoh, apabila terdapat 10 individu di awal generasi, maka jumlah tersebut harus dijaga tetap sama (i.e., 10 individu) hingga akhir generasi. Mekanisme seleksi inilah yang membuat generasi terakhir tetap sama, yang diharapkan merupakan generasi terbaik dengan informasi genetika terbaik yang ditandai dengan *fitness value* yang paling besar.

Bagian ini memberikan ilustrasi penggunaan dari roda rolet untuk proses seleksi. Dalam roda rolet, setiap individu dalam populasi memiliki ruang pada roda rolet proporsional dengan *fitness value*-nya. Tabel 10.1 memberikan ilustrasi proses menentukan ukuran ruang pada roda rolet untuk masing-masing individu. Pada 10.1 terlihat beberapa individu dengan informasi genetika yang berbeda-beda pada *chromosome*. Sehingga setiap individu memiliki *fitness value* yang berbeda-beda pula. Nilai

fitness ini kemudian digunakan untuk menentukan proporsi ruang pada roda rolet untuk masing-masing individu. Prosesnya dapat dilakukan dengan menjumlahkan seluruh nilai *fitness* yang kemudian digunakan sebagai pembagi nilai *fitness* pada masing-masing individu. Hasil pembagian ini kemudian digunakan untuk menghitung persentase ruang pada roda rolet. Individu dengan *fitness value* yang tinggi akan memiliki proporsi yang besar, sehingga memiliki peluang untuk terpilih lebih besar. Sedangkan individu yang memiliki *fitness value* yang kecil memiliki kemungkinan yang kecil untuk terpilih. Proses pemilihan kemudian dilakukan secara acak sebanyak jumlah individu dalam satu generasi. Proses pemilihan secara acak memungkinkan pula adanya variasi dari informasi genetika yang ada didalam populasi. Variasi ini berperan dalam menjaga GA untuk tidak terjebak di dalam lokal optimal.



Gambar 10.3 Contoh roda rolet.

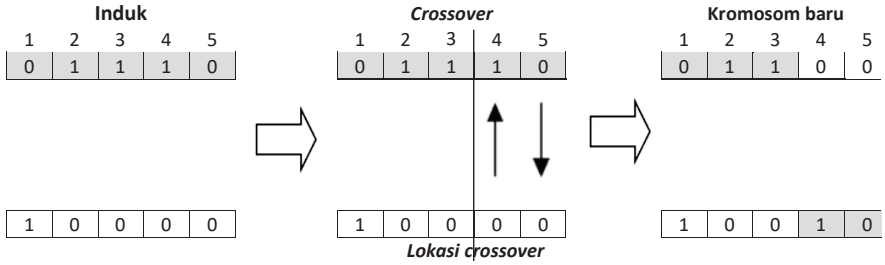
Tabel 10.1 Contoh % *fitness value* untuk roda rolet.

No	Chromosome	*Fitness value*	%Total
1	00101110	6.82	31
2	00110010	1.11	5
3	10100011	8.48	38
4	01100100	2.57	12
5	10111001	3.08	14
Total		**22.06**	**100**

9.2.2. *Operator Crossover*

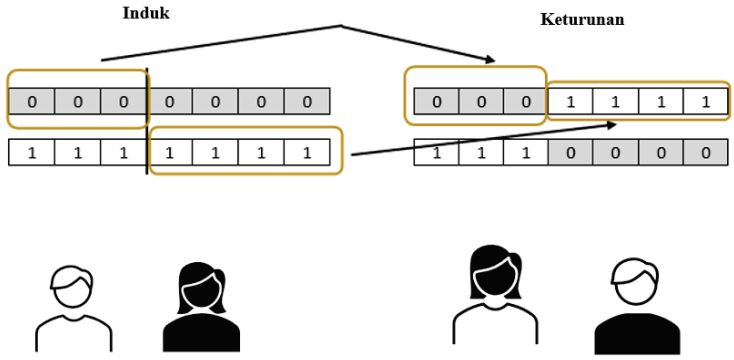
Selain operator seleksi, operator dasar lain dari GA adalah *crossover,* yan*g* berusaha merepresentasikan proses kawin silang untuk menghasilkan keturunan. *Crossover* menghasilkan individu baru dengan cara melakukan kawin silang informasi genetik dalam dari kromosom induk generasi sebelumnya. Induk yang akan dikawinkan dipilih secara acak dari individu dalam populasi, kemudian informasi genetika kedua induk disilangkan

secara acak. Proses ini kemudian menghasilkan keturunan baru hasil persilangan, yang menggabungkan informasi genetika yang ada di kedua induk. Gabungan informasi genetika ini diharapkan memiliki *fitness value* yang lebih baik dibandingkan kedua induknya, meskipun tidak menutup kemungkinan *fitness value*-nya lebih buruk. Proses kawin silang dapat dilakukan dengan banyak cara, meskipun salah satu yang paling sederhana adalah dengan pendekatan *single location* yang diilustrasikan pada Gambar 10.4 dan 10.5



Gambar 10.4 Proses *single crossover.*

Hasil dari *crossover* di atas akan menghasilkan 2 individu baru yang baru yang mewarisi informasi genetika dari kedua induknya. Informasi yang dimiliki bisa sama dengan induknya, ataupun dapat berbeda, yang menyebabkan *fitness value* nya berbeda pula dari kedua induknya.

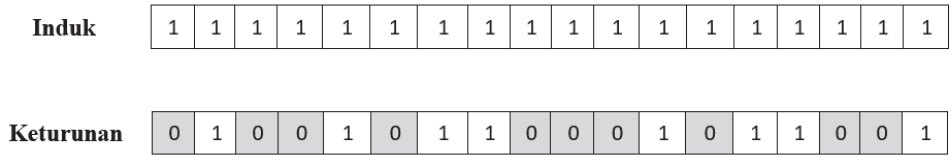


Gambar 10.5 Ilustrasi *crossover.*

9.2.3. Operator Mutasi

Operator mutasi merepresentasikan proses mutasi genetik yang umum terjadi dalam proses genetika makhluk hidup. Operator ini memiliki karakteristik yang sama dengan yang terjadi pada proses genetika alam, yang selalu melibatkan perubahan secara signifikan. Sehingga informasi genetik pada sebuah individu dapat sangat berbeda dengan informasi genetik yang ada di induknya. Pada GA, operator mutasi akan menentukan secara

acak lokasi *gen* yang akan bermutasi kemudian nilai *alle* pada *gen* tersebut berubah. Sebagai contoh dalam konteks biner, jika nilai *allele* adalah 1 maka setelah dimutasi akan menjadi 0 (lihat Gambar 10.6). Mutasi memiliki peran signifikan dalam rangka melakukan eksplorasi pada ruang pencarian dan menghindarkan GA dari solusi lokal optimal.



Gambar 10.6 Terjadinya mutasi.

10.2. Simple Genetic Algorithm (SGA)

Pada buku ini, jenis GA yang akan dibahas prosedur implementasinya adalah *Simple* GA. SGA merupakan versi dari GA yang hanya memiliki operator utama saja, yaitu *cross over*, *mutation*, dan *selection*. Adapun prosedur untuk mengaplikasi SGA dapat dijelaskan sebagai berikut:

Langkah 1. Inisialisasi

- Tetapkan generasi (g)=0
- Tentukan jumlah individu dalam satu populasi dan bangkitkan individu dalam populasi awal secara acak.
- Tentukan rate dari operator *crossover* dan mutasi

Langkah 2. Perhitungan *fitness value*

- Hitung nilai *fitness* dari *objective function* yang telah ditetapkan sebelumnya, dengan menggunakan informasi genetika (i.e., kromosom) pada setiap individu.

Langkah 3: *Crossover* (*Single Point*)

- Bangkitkan bilangan acak dari nol hingga 1 untuk menentukan apakah akan dilakukan proses *crossover*. Jika bilangan acak lebih kecil dari *crossover rate*, maka lakukan *crossover* jika tidak lanjutkan ke **Langkah 4**.
- Jika bilangan acak lebih kecil dari *crossover rate*, maka pilih dua individu yang akan dikawinkan secara acak. Setelah proses perkawinan akan terdapat dua individu baru hasil *crossover*.

Langkah 4: *Mutation*

- Bangkitkan bilangan acak dari nol hingga 1 untuk menentukan apakah akan dilakukan proses mutasi. Jika bilangan acak lebih kecil dari *mutation rate*, maka lakukan mutasi, jika tidak lanjutkan ke **Langkah 5**.

- Jika bilangan acak lebih kecil dari *mutation rate*, maka pilih individu yang akan dimutasi secara acak. Setelah itu pilih lokasi untuk proses mutasi, sehingga akan diperoleh satu individu baru hasil mutasi.

Langkah 5: *Selection*

- Pada tahap ini, seluruh individu mulai dari individu generasi sebelumnya, keturunan hasil kawin silang dan mutasi akan di seleksi. Proses ini digunakan untuk menentukan yang akan bertahan pada generasi selanjutnya yang dengan roda rolet.
- Roda rolet diputar sebanyak jumlah individu dalam satu generasi.
- Tambahkan $g=g+1$, jika kriteria berhenti terpenuhi (misal jumlah generasi maksimum), maka hentikan proses SGA. Jika jumlah generasi masih kurang dari jumlah generasi maksimum, maka diulangi dari **Langkah 2**.

Langka-langkah di atas dilakukan secara iteratif hingga akhir dari generasi. Mekanisme seleksi kemudian menghasilkan keturunan terkadang disebut juga sebagai mekanisme reproduksi. Individu yang lolos seleksi akan masuk ke tahap reproduksi yang kemudian akan melahirkan individu yang baru untuk melanjutkan keturunan. Individu pada generasi terakhir ini diharapkan merupakan generasi terbaik yang lolos tahap seleksi dan telah mewarisi informasi genetika terbaik dari induknya.

10.3. Aplikasi GA dengan *Ms. Excel*

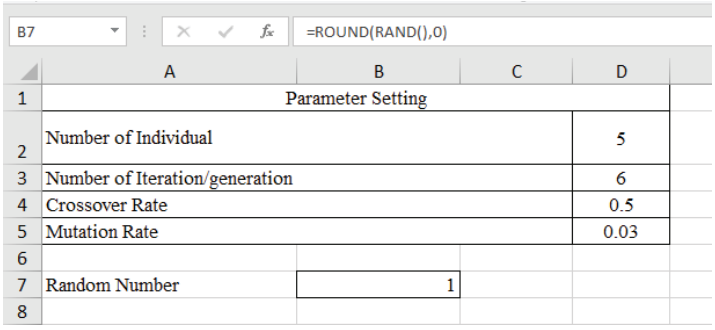
Pada bagian ini GA akan diaplikasikan dengan menggunakan *Ms. Excel* untuk menyelesaikan permasalahan seperti yang disampaikan pada sub bab 9.8. Untuk menyelesaikan permasalahan tersebut berikut beberapa langkah-langkahnya.

- **Langkah 1 & 2:** identik dengan sub bab 9.8, kecuali istilah *bit* pada GA menjadi *allele*
- **Langkah 3**: tentukan parameter terkait jumlah individu dalam populasi, jumlah generasi, *crossover rate, mutation rate.* Contoh ini diberikan pada Tabel 10.2

Tabel 10.2 *Parameter settings GA.*

Jumlah Individu	5
Jumlah Generasi	6
Crossover Rate	0.5
Mutation Rate	0.03

- **Langkah 4**: inisialisasi *allele* dengan menggunakan bilangan acak 0 atau 1, dengan menggunakan fungsi *round (rand)* pada *Ms. Excel* lihat Gambar 10.8

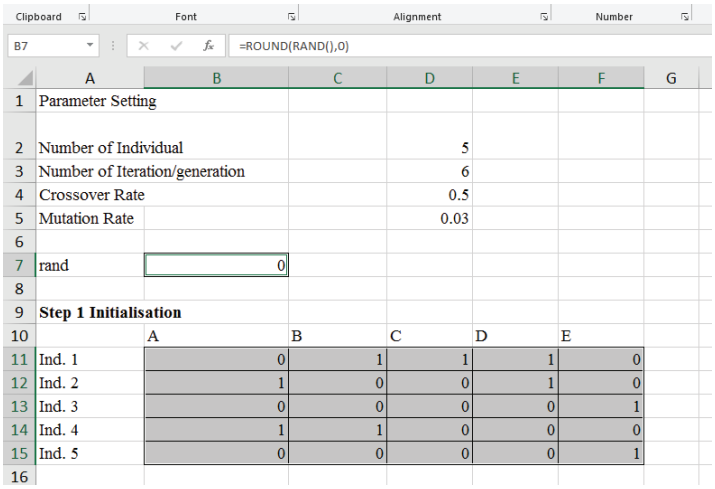


B7 =ROUND(RAND(),0)

	A	B	C	D
1	Parameter Setting			
2	Number of Individual			5
3	Number of Iteration/generation			6
4	Crossover Rate			0.5
5	Mutation Rate			0.03
6				
7	Random Number	1		
8				

Gambar 10.7 Rumus bilangan acak pada *Ms. Excel.*

Fungsi tersebut kemudian di *copy +paste* di setiap *cell* pada tabel awal, maka hasilnya dapat dilihat pada Gambar 10.8.



Clipboard Font Alignment Number

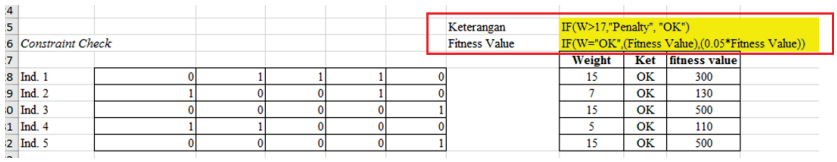
B7 =ROUND(RAND(),0)

	A	B	C	D	E	F	G
1	Parameter Setting						
2	Number of Individual			5			
3	Number of Iteration/generation			6			
4	Crossover Rate			0.5			
5	Mutation Rate			0.03			
6							
7	rand	0					
8							
9	**Step 1 Initialisation**						
10		A	B	C	D	E	
11	Ind. 1	0	1	1	1	0	
12	Ind. 2	1	0	0	1	0	
13	Ind. 3	0	0	0	0	1	
14	Ind. 4	1	1	0	0	0	
15	Ind. 5	0	0	0	0	1	
16							

Gambar 10.8 Hasil dari *copy+paste* rumus bilangan acak.

- **Langkah 5**: Evaluasi *fitness value* dan fungsi kendala

  Evaluasi umumnya dilakukan dengan memasukkan *decision variable* ke dalam *objective function* yang dikenal sebagai *fitness value*. Perhitungan *fitness value* dapat menggunakan Persamaan (9.40), sementara untuk mengevaluasi apakah kandidat solusi melebihi kendala kapasitas yang ditetapkan gunakan Persamaan (9.41). Jika kandidat solusi memiliki berat yang lebih besar daripada batasan kapasitas maka *fitness value* akan dipenalti dengan menggalikan *fitness value* nya dengan 0,05. Resume dari penerapannya dapat dilihat pada Gambar 10.9 dan Tabel 10.3



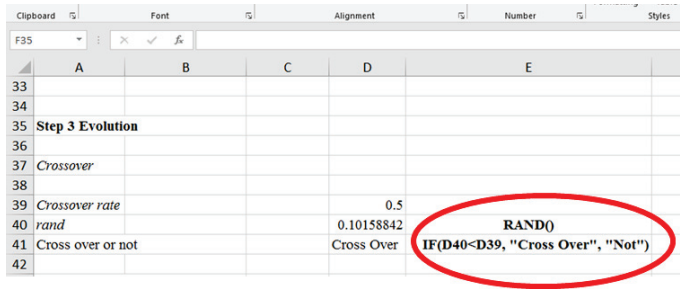
Gambar 10.9 Evaluasi *fitness value* GA.

Tabel 10.3 Hasil *fitness value* setelah pengecekan batasan kapasitas.

	Berat	Keterangan Penalti	*fitness value*
Individu 1	15	OK	300
Individu 2	7	OK	130
Individu 3	15	OK	500
Individu 4	5	OK	110
Individu 5	15	OK	500

- **Langkah 6**: Lakukan perkawinan silang

  Proses perkawinan silang dilakukan dengan memasangkan individu untuk ditukarkan informasi genetikanya. Proses ini dilakukan dengan mencari terlebih dahulu apakah akan terjadi perkawinan silang. Proses ini dilakukan dengan melihat apakah bilangan acak yang dibangkitkan dari *Ms. Excel* lebih kecil dari *crossover rate*. Jika kecil maka proses perkawinan silang akan dilakukan (Lihat Gambar 10.10 sebagai ilustrasi).



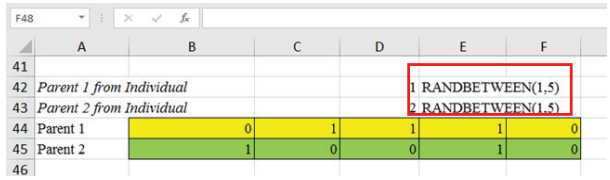
Gambar 10.10 Rumus di *Ms. Excel* untuk menentukan terjadi kawin silang

Pada penanda lingkaran merah merupakan keterangan penggunaan rumus pada aplikasi *Excel* untuk mencari apakah akan terjadi *crossover* atau tidak. Pada contoh ini diasumsikan akan terjadi *crossover* (lihat Tabel 10.4)

Tabel 10.4 Contoh hasil *crossover.*

Crossover rate	0.5
rand	0.10158842
Cross over or not	*Cross Over*

Setelah mendapatkan keterangan bahwa telah terjadi *crossover* maka selanjutnya dilakukan penentuan induk dari hasil *crossover* menggunakan nilai acak terhadap 5 individu sebelumnya. Cara penentuan *parents* dapat dilihat pada Gambar 10.11. Penanda merah merupakan rumus pencarian *parent* dengan *random number.*



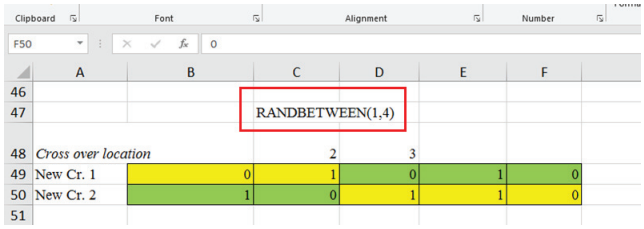
Gambar 10.11 Rumus di *Ms. Excel* untuk menentukan *parent* dari *crossover.*

Dari nilai random tersebut di dapat *parent 1* merupakan individu 1, sedangkan *parent 2* merupakan individu 2, sehingga hasilnya dapat dilihat pada Gambar 10.12.

Parent 1	0	1	1	1	0
Parent 2	1	0	0	1	0

Gambar 10.12 *Parent* yang terpilih.

Untuk lokasi persilangan dilakukan juga dengan menggunakan nilai *random* seperti terlihat pada Gambar 10.13.



Gambar 10.13 Penentuan lokasi persilangan.

Pada hasil nilai *random,* didapat persilangan dilakukan antara 2 tempat *parent* 1 dan 3 tempat *parent* 2, maka didapat pembentukan individu baru seperti pada Gambar 10.14.

Individu hasil *crossover* 1	0	1	0	1	0
Individu hasil *crossover* 2	1	0	1	1	0

Gambar 10.14 Dua individu baru hasil *crossover.*

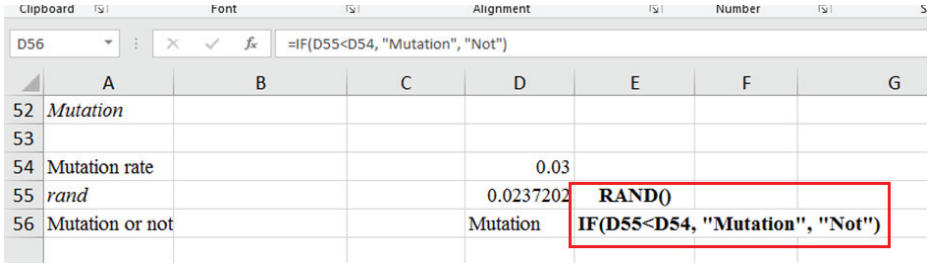
Setelah mendapatkan individu yang baru, maka lakukan kembali perhitungan tentang berat dan *fitness value* dari individu yang baru. Berat dan *fitness value* dari individu yang baru dapat dilihat pada Tabel 10.5.

Tabel 10.5 Berat dan *fitness value* individu baru.

Individu	**Berat**	**Keterangan Penalti**	***fitness value checked***
Individu hasil crossover 1	8	OK	180
Individu hasil crossover 2	14	OK	250

- **Langkah 7**: Lakukan mutasi

  Tahapan selanjutnya dilakukan dengan mengecek apakah akan terjadi proses mutasi dengan mendasarkan kepada *mutation rate* yang telah ditetapkan sebelumnya. Proses mutasi dilakukan jika bilangan acak yang dibangkitan lebih kecil daripada *mutation* rate. Proses membangkitkan bilangan acak dilakukan dengan fungsi rand() pada *Ms. Excel* (lihat Gambar 10.15).



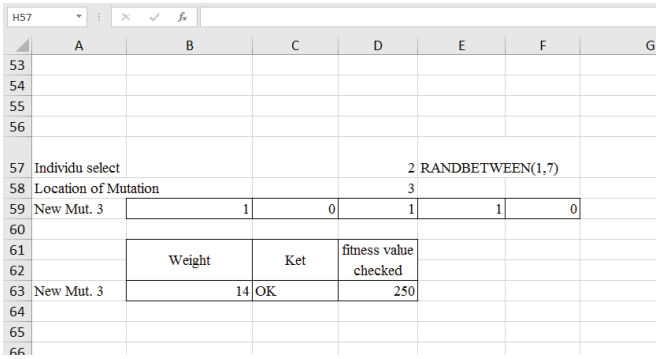
Gambar 10.15 Rumus di *Ms. Excel* penentuan mutasi.

Untuk tahap ini anggap terjadi proses mutasi yang ditandai oleh nilai *random* yang lebih kecil dari *mutation rate* 0.03 (lihat Tabel 10.6)

Tabel 10.6 Contoh hasil penentuan mutasi.

Mutation rate	0.03
rand	0.0237202
Mutation or not	*Mutation*

Setelah diketahui akan terjadi mutasi, proses dilanjutkan dengan memilih individu yang akan dimutasi. Pencarian individu yang terpilih mutasi menggunakan rumus *randbetween* (1, (*jumlah individu+jumlah individu baru*)). Sebagai ilustrasi pada kasus ini, terdapat 5 individu awal yang sudah bertambah 2 individu baru ketika melakukan *crossover.* Sehingga terdapat 7 individu yang mungkin untuk bermutasi. Pada contoh ini, anggaplah individu nomor 2 yang mengalami mutasi dan lokasi *allele* yang mengalami perubahan yaitu pada lokasi *allele* ke-3. Pada lokasi ini sebelumnya bernilai 0, sehingga ketika terjadi mutasi akan menganti nilainya menjadi 1, sehingga hasilnya dapat dilihat pada Gambar 10.16.



	A	B	C	D	E	F	G
53							
54							
55							
56							
57	Individu select			2	RANDBETWEEN(1,7)		
58	Location of Mutation			3			
59	New Mut. 3	1	0	1	1	0	
60							
61		Weight	Ket	fitness value checked			
62							
63	New Mut. 3	14	OK	250			
64							
65							
66							

Gambar 10.16 Contoh hasil individu bermutasi.

Berdasarkan hasil dari *crossover* dan *mutation,* maka individu yang bertambah menjadi 3 individu (i.e., 8 individu dalam populasi), sehingga langkah selanjutnya dalam melakukan proses seleksi agar jumlah individu dalam populasi tetap 5 individu.

- **Langkah 8**: Lakukan proses seleksi

  Pada tahap ini akan dilakukan proses seleksi untuk mencari 5 individu yang akan melanjutkan pada generasi selanjutnya. Setiap individu yang didapat sebelumnya (i.e., 5 individu awal, 2 individu baru dari *crossover,* 1 individu baru dari *mutasi*) akan diseleksi dengan mendasarkan kepada *fitness value*-nya untuk menghindari tidak adanya variasi karena hanya mengutamakan individu dengan *fitness value* terbesar yang dapat melanjutkan ke generasi selanjutnya.

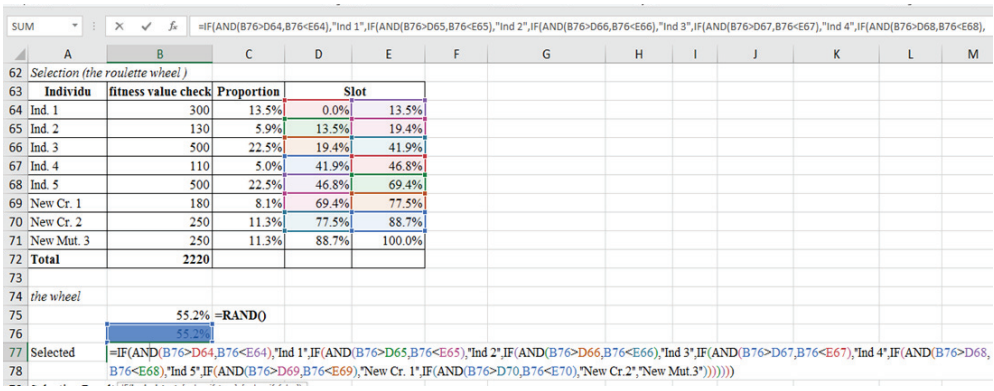
  Pada buku ini digunakan pendekatan *roulette wheel*, yang mengkonsiderasikan tidak hanya *fitness value* akan tetapi mempertimbangkan pula unsur keacakan dalam memilih individu yang akan melanjutkan ke generasi selanjutnya. Unsur keacakan ini menjadi penting dalam rangka menjaga variasi dari informasi genetika yang ada di dalam individu.

Tabel 10.7 Contoh penggunaan roda rolet.

Individu	***fitness value* check**	**Proportion**	**Slot**	
Ind. 1	300	13.5%	0.0%	13.5%
Ind. 2	130	5.9%	13.5%	19.4%
Ind. 3	500	22.5%	19.4%	41.9%
Ind. 4	110	5.0%	41.9%	46.8%
Ind. 5	500	22.5%	46.8%	69.4%
New Cr. 1	180	8.1%	69.4%	77.5%
New Cr. 2	250	11.3%	77.5%	88.7%
New Mut. 3	250	11.3%	88.7%	100.0%
Total	**2220**			

Pendekatan roda rolet secara sederhana dapat dilakukan dengan membuat semacam proporsi pada setiap individu berdasarkan *fitness value*-nya (lihat Persamaan (10.1)). Proporsi ini kemudian akan digunakan untuk membuat slot pada masing-masing individu. Kemudian dibuat interval untuk masing-masing individu dengan mengurutkan secara kumulatif. Sehingga mendapatkan interval seperti yang dapat dilihat pada Tabel 10.7.

Proses selanjutnya dilakukan dengan membangkitkan bilangan acak, dimana bilangan acak tersebut akan ditempatkan kepada slot-slot yang telah ditentukan sebelumnya (lihat Gambar 10.17). Individu yang bilangan acak berada pada slot tertentu, kemudian dipilih sebagai individu yang akan melanjutkan ke generasi selanjutnya. Proses ini diulang sebanyak jumlah individu yang telah ditetapkan sebelumnya (i.e., 5 individu dalam kasus ini). Hasil dari proses seleksi ini dapat dilihat pada Tabel 10.8.



Gambar 10.17 Rumus di *Ms*. *Excel* untuk seleksi individu.

Tabel 10.8 Individu yang melanjutkan ke generasi selanjutnya.

Individu 1	1	0	0	1	0
Individu 2	0	0	0	0	1
Individu 3	0	1	0	1	0
Individu 4	0	0	0	0	1
Individu 5	1	0	1	1	0

- **Langkah 9**: Perbaharui nilai generasi (g=g+1) (i.e., iterasi), jika generasi lebih besar dari jumlah generasi maksimal yang ditetapkan, maka hentikan proses perhitungan. Jika tidak, maka ulangi kembali dari **Langkah 6**.

Template *Ms. Excel* untuk GA dapat diperoleh melalui laman berikut:

https://*bit*.ly/GAtemplatebuku

Pembaca yang tertarik untuk melihat penjelasan dalam bentuk audio-visual terkait Bab ini dapat mengunjungi playlist youtube berikut:

https://bit.ly/Playlist-GA

11. *Glowworm Swarm Optimization* (GSO)

Glowworm Swarm Optimization (GSO) merupakan salah satu varian dari *metaheuristik* yang dikembangkan oleh (Krishnanand, and Ghose, 2005;2008) untuk menangani permasalahan optimasi dalam bentuk kontinu. Zukhruf et al (2020) mengembangkan GSO untuk permasalahan binari yang dibahas dalam buku ini.

11. 1. Prosedur Glowworm Swarm Optimization

GSO terinspirasi dari perilaku cacing yang memiliki pendar cahaya (i.e., *glowworm*). Serupa dengan PSO, posisi dari *glowworm* dalam ruang pencarian kemudian menentukan nilai *fitness* dari setiap *glowworm*. Sehingga GSO berusaha memandu setiap *glowworm* untuk mencari posisi terbaik yang memberikan nilai *fitness* terbaik. Meskipun dalam konteks pertukaran informasi, GSO menggunakan variabel *luciferin*.

Jenis hewan ini umumnya menarik *glowworm* lainnya dengan iluminasinya, dimana *glowworm* yang lebih baik cahayanya akan menarik lebih kuat. Dengan mendasarkan kepada fenomena natural ini, GSO menggunakan istilah *luciferin* yang menggambarkan tingkat iluminasi *glowworm*. Kemudian nilai *luciferin* ini dikorelasikan dengan nilai *fitness*-nya atau nilai fungsi tujuannya. Dimana *luciferin* yang tinggi akan menyebabkan *glowworm* memiliki pendar cahaya yang tinggi. *Luciferin* yang tinggi kemudian akan memberikan daya tarik lebih luas kepada *glowworm* lainnya. Sehingga potensi *glowworm* lain akan menuju kepada arah *glowworm* tersebut akan semakin tinggi. Pembaharuan posisi dari setiap *glowworm* akan didasarkan kepada arah pergerakan sebelumnya dan arah pergerakan *glowworm* lain. Dimana arah pergerakan yang terpengaruh oleh *glowworm* lain didasarkan kepada probabilitas perpindahan yang merupakan fungsi dari nilai *luciferin*.

Adapun prosedur umum dari penerapan GSO (Zukhruf et al, 2020) adalah sebagai berikut:

Tahap 1. Inisialisasi (t=0)

- Tentukan nilai awal dari beberapa parameter utama berikut yaitu, jumlah *glowworm* dalam *swarm* (N), jumlah iterasi (*iter*), nilai konstanta terkait *luciferin* (ρ danγ), nilai

konstanta terkait daya jangkau (β dan o), nilai konstanta terkait *stepsize* posisi (s), probabilitas maksimum dan minimum perubahan posisi ($prob_{min}$dan $prob_{max}$), dan nilai maksimum daya jangkau (r_{max})

- Tentukan posisi awal dengan menggunakan bilangan random (x_i^t) dimana *i = 1, 2, . . . ,N*
- Tentukan nilai awal *luciferin* (l_i^t) dan daya jangkaunya (r_i^t) untuk setiap *glowworm i* = 1, 2, . . . ,*N*
- Berdasarkan posisi *glowworm* (x_i^t), hitung jarak dari *glowworms i* ke *glowworms j* ($d1_{ij}^t$)
- Hitung probabilitas berpindah kepada posisi dari *glowworm* terdekat untuk setiap *glowworm i,* menggunakan fungsi berikut:

$$b_{ij}^t = \frac{l_i^t - l_j^t}{\sum_{k \in N_i^t} l_k^t - l_j^t} \tag{11.1}$$

dimana $j \in N_i^t, N_i^t = \{j: d1_{ij}^t < r_i^t, l_i^t < l_j^t\}$ adalah kumpulan glowworm yang berdekatan dengan glowworm of i pada iterasi-t

- Tentukan arah dari perpindahan atau pergerakan dengan mengikuti probabilitas berpindah tertinggi, kemudian perbaharui posisi *glowworm* berdasarkan formula berikut:

$$x_i^{t+1} = x_i^t + s\left(\frac{x_j^t - x_i^t}{\left\|x_j^t - x_j^t\right\|}\right) \tag{11.2}$$

Dimana *s* (>0) adalah ukuran berpindah (i.e., *stepsize*)

Tahap 2. Tentukan posisi *glowworm* dalam bentuk biner dengan menggunakan fungsi probabilitas berikut:

$$prob_{ih} = \frac{(x_{ih}^t - prob_{min})}{(probmin_{max})} \tag{11.3}$$

$$prob_{ih} = \frac{(x_{ih}^t - prob_{min})}{(probmin_{max})} \tag{11.4}$$

$$u_{ih}^{t+1} = \begin{cases} 1 \; if \; rand \leq prob_{ih} (0 \leq prob_{ih} \leq 1) \\ 0 \; else \end{cases} \tag{11.5}$$

dimana, $u_i^{t+1} = [u_{i1}^{t+1}, u_{i2}^{t+1}, u_{i3}^{t+1}, \ldots, u_{ih}^{t+1}, \ldots . u_{iH}^{t+1}]$ and rand adalah bilangan acak antara 0 hingga 1

Tahap 3. Hitung nilai *fitness* masing-masing *glowworm* (z_i^{t+1}), dan perbaharui nilai *luciferin* dengan menggunakan formula berikut:

$$l_i^{t+1} = (1-\rho)\ l_i^t + \gamma z(u_i^{t+1}) \quad (11.6)$$

dimana ρ dan γadalah konstanta yang merepresentasikan penurunan dan kenaikan nilai dari *luciferin*.

Tahap 4. Perbaharui daya jangkau dari masing-masing *glowworm* (r_i^t),

$$r_i^{t+1} = min\{r\ max\{0, r_i^t + \beta(o - |N_i^t|)\}_{max}\ \{\}\} \quad (11.7)$$

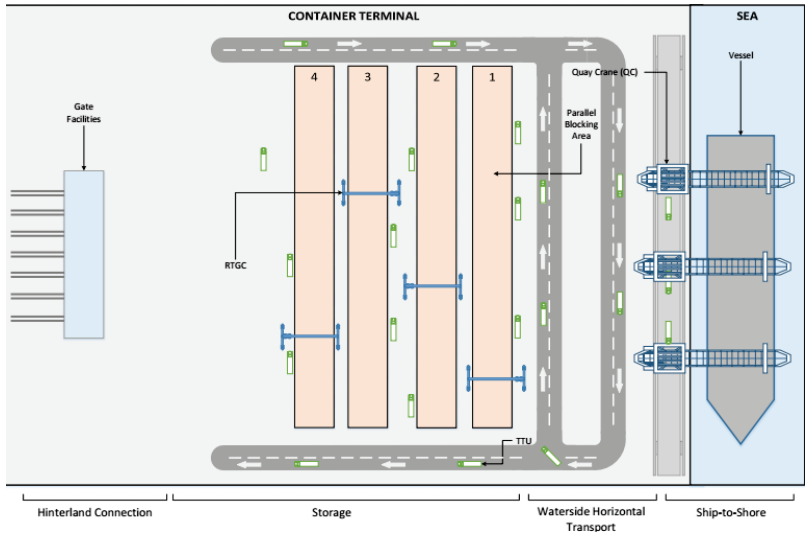
dimana, r_{max} adalah nilai maksimum dari r_i^t, dan o menunjukkan konstanta untuk membatasi jumlah *glowworm* dalam daya jangkaunya.

Tahap 5. Perbaharui jarak dan hitung probabilitas berpindah seperti pada Persamaan (11.1)

Tahap 6. Hitung t = t+1. Hentikan proses perhitungan jika kriteria berhenti telah terpenuhi, jika belum Kembali ke **Tahap 2**.

11. 2. Contoh Aplikasi GSO

Bagian ini memberikan contoh dari aplikasi GSO dalam menyelesaikan permasalahan optimasi untuk mendesain fasilitas terminal kontainer (CT). Sistem operasi yang dikonsiderasikan dimulai dari barang di kapal hingga berakhir di lapangan kontainer (CY). Sehingga desain yang dikonsiderasikan adalah jumlah unit truk (TTU), dan *gantry crane* (RTGC) yang dibutuhkan pada lapangan penumpukan, yang ilustrasinya dapat lihat Gambar 11.1.



Gambar 11.1 Proses di CT yang dikonsiderasikan (Zukhruf et al, 2020).

Untuk penanganan kapal dan peti kemasnya, CT dilayani oleh 3 dermaga dengan 3-unit *quay crane* (QC) di setiap dermaga yang dimungkinkan untuk dioperasikan secara bersamaan. Di sisi darat, CT dilengkapi masing-masing 45-unit dan 10-unit TTU dan RTGC. Dari sisi permintaan, jenis kapal dan frekuensi yang dikunjungi setiap dermaga bervariasi mengikuti distribusi tertentu.

Sebelum membahas lebih dalam tentang masalah optimasi desain CT, penting untuk mempresentasikan kondisi awal dari kinerja CT. Pada kondisi dasar peti kemas membutuhkan rata-rata 33,63 jam untuk tiba dan menumpuk di lapangan peti kemas. Permasalahan optimasi kemudian dapat dilihat dalam kerangka menentukan jumlah peralatan dan kombinasinya secara optimal (yaitu, TTU dan RTGC). Karena total waktu tempuh yang terjadi dipengaruhi oleh produktivitas peralatan, peningkatan jumlah peralatan diharapkan mengarah pada pengurangan waktu total. Namun, karena kenaikan TTU mungkin membawa peningkatan penundaan di jalan, menjadi penting untuk mencari jumlah TTU untuk mengurangi penundaan.

Teknik solusi berbasis *swarm* kemudian dipanggil untuk menentukan jumlah fasilitas yang optimal (yaitu, TTU dan RTGC). Karena fungsi tujuan mempertimbangkan rasio manfaat dan biaya, maka nilai waktu untuk setiap kontainer ditetapkan Rp 1,95 juta per TEUs per jam, biaya pembelian peralatan ditetapkan sebesar Rp 500 juta per unit, Rp 1.500 juta per unit, untuk TTU dan RTGC, masing-masing. Selanjutnya, diasumsikan bahwa permintaan bulanan selalu sama selama tahun operasi (yaitu, 5 tahun).

Berdasarkan hasil optimasi GSO, tindakan optimal untuk meningkatkan kinerja CT meliputi penambahan masing-masing 11 dan 5-unit TTU dan RTGC. Tindakan peningkatan ini secara langsung mengurangi total waktu perjalanan untuk tiba di CY, di mana dapat berkurang hingga 5%.

11. 3. Perbandingan Performa GSO dan PSO

Kinerja algoritma *swarm* kemudian dievaluasi berdasarkan hasil optimasi dan waktu berjalannya. Hasil optimasi didefinisikan sebagai nilai *fitness* dari fungsi tujuan, yang dinilai dari 10 *run* berdasarkan nilai solusi maksimum, rata-rata, dan minimum. Untuk memastikan perbandingan yang sebanding, jumlah solusi yang mungkin ditetapkan sebesar 4500, yang secara praktis digunakan dalam penelitian optimasi sebelumnya (Yamada and Zukhruf, 2015)

Perbandingan kinerja GSO, PBPSO dan MPBPSO disajikan pada tabel di bawah ini, yang menyimpulkan bahwa GSO dapat memberikan hasil yang lebih baik daripada algoritma berbasis *swarm* lainnya berdasarkan nilai terbaik dan rata-rata. Selain itu, hasil MPBPSO menawarkan hasil terbaik yang lebih baik daripada PBPSO. Namun, seperti yang ditunjukkan oleh Tabel 11.1, semua algoritma *swarm* masih menghadapi masalah stabilitas untuk mengatasi masalah optimasi, di mana mereka tidak memberikan hasil kualitas yang sama dalam 10 kali berjalan. Waktu komputasinya juga dibandingkan, dimana dinilai dengan PC dengan CPU Intel Core i5 2,2 GHz dan RAM 16,0 GB, GSO menjanjikan waktu komputasi tercepat, yang diikuti oleh MPBPSO. Berdasarkan hasil ini, dapat disimpulkan bahwa GSO versi biner berpotensi untuk diterapkan untuk mengatasi masalah optimasi, meskipun masalah stabilitasnya perlu ditingkatkan.

Tabel 11.1 Perbandingan kinerja GSO dan PSO.

	GSO	***MPBPSO***	***PBPSO***
Best	1.01	1.00	0.94
Average	0.90	0.85	0.84
Worst	0.73	0.71	0.74
Computation Time (sec)	8,372	14,264	17,345

12. Metode Penyelesaian Masalah Optimasi Lainnya

12.1. Full enumerasi

Permasalahan optimasi pada intinya adalah seni memilih keputusan dari begitu banyak kemungkinan keputusan. Sehingga permasalahan ini terkadang dianggap sebagai seni untuk "menebak (*guessing*)". Untuk memperoleh hasil yang paling optimal (i.e., global optimal), pada dasarnya kita dapat menguji semua kemungkinan yang ada. Dimana keputusan yang memberikan nilai maksimal atau minimal merupakan solusi optimal. Sebagai contoh, anggaplah kita menghadapi permasalahan optimasi terkait pemilihan rute untuk meminimalkan waktu tempuh. Untuk mencapai lokasi tujuan, pengemudi harus melewati dua buah simpang, dimana terdapat pilihan 3 buah simpang dengan waktu tempuh antar simpang dapat dilihat pada Tabel 12.1 sebagai berikut:

Tabel 12.1 Waktu tempuh antar simpang (menit).

Simpang	A	B	C	LA	LT
A	-	1	2	2	2
B	1	-	3	3	1
C	2	3	-	1	3
LA	2	3	1	-	-
LT	2	1	3	-	-

LA: Lokasi Asal; LT: Lokasi Tujuan

Dengan menganggap bahwa pengemudi harus bergerak dari **LA** menuju **LT** dengan melewati dua buah simpang, maka pengemudi dapat menebak rute yang dipilih, sebagai contoh:

LA-B-C-LT dengan waktu tempuh 9 menit (3+3+3).

Untuk medapatkan waktu tempuh yang paling minimal, pengemudi pada dasarnya tinggal mengevaluasi semua pilihan yang mungkin yang dapat diresumekan sebagai berikut:

Tabel 12.2 Enumerasi pilihan rute.

No	Urutan	Waktu
1	LA-A-B-LT	4
2	LA-A-C-LT	7
3	LA-C-B-LT	5
4	LA-B-A-LT	6
5	LA-C-A-LT	5
6	LA-B-C-LT	9

Tabel 12.2 memberikan ilustrasi terkait evaluasi waktu tempuh untuk seluruh keputusan pilihan rute yang mungkin. Dapat terlihat bahwa rute yang melewati urutan simpang A dan B memberikan waktu tempuh terrendah. Sehingga dapat disimpulkan bahwa solusi tersebut merupakan solusi paling optimal, karena seluruh kemungkinan yang ada telah dievaluasi. Proses ini dapat disebut sebagai metode enumerasi sempurna. Metode ini mendasarkan kepada proses evaluasi seluruh kemungkinan keputusan yang ada, meskipun metode ini hanya cocok untuk jumlah keputusan berjumlah kecil. Jika jumlah pilihan besar, maka metode ini sangat sulit untuk diterapkan (lihat kompleksitas masalah pada bab 1 di buku ini).

12.2. *Djikstra*

Masalah lain yang umumnya diselesaikan dengan metode eksak adalah masalah pencarian rute terpendek. Terdapat berbagai metode untuk mencari rute terpendek ini, meskipun salah satu metode yang cukup banyak digunakan di dalam dunia transportasi adalah metode *Djikstra*. Metode ini diambil dari inventornya yaitu *Edsger W. Dijkstra* yang merupakan seorang cendikiawan ilmu komputer dari Belanda. Untuk menjalankan metode *Djikstra* terdapat 4 tahapan utama yaitu:

Tahap 1: Tetapkan semua *node* yang ada dengan label '**non-permanen**'

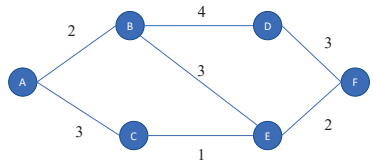
Tahap 2: Tetapkan satu *node* sebagai *node* awal (***source***), serta tetapkan sebagai *node* pertama yang diberikan label **permanen**. *Node* awal ini juga ditetapkan sebagai *node* aktif.

Tahap 3: Periksa semua *node* yang **non-permanen-*i*** yang berdekatan dengan *node* aktif.

- Untuk setiap ***node* non-permanen-*i*** hitung jarak kumulatif dari *node* awal (***source***) ke ***node* non-permanen-*i*** dengan melalui *node* aktif
- Perbaharui nilai jarak berdasarkan nilai jarak terpendek dari ***source***
- Dari semua **non-permanen-*i*** yang diperiksa pilih satu *node* dengan nilai jarak terkecil, dan tetapkan sebagai ***node* permanen**
- Tetapkan ***node* permanen-*i*** sebagai *node* aktif

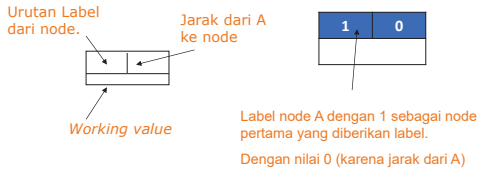
Tahap 4: Ulangi proses pada Tahap 3 hingga semua *node* sudah berganti sebagai ***Node* Permanen**

Untuk memberikan pemahaman lebih baik terhadap metode ini, contoh dalam mencari rute terbaik diberikan dengan menggunakan jaringan sederhana seperti terlihat pada Gambar 12.1.

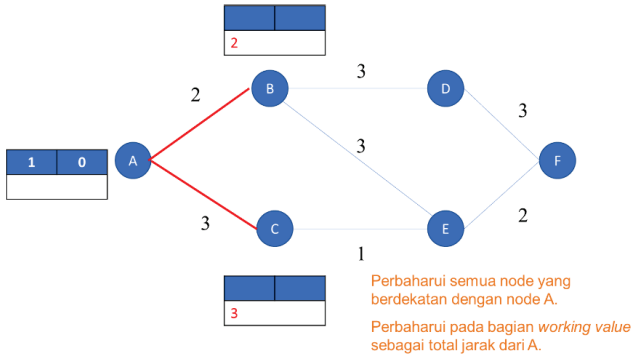


Gambar 12.1 Contoh jaringan sederhana untuk kasus pencarian rute terbaik.

Jaringan tersebut memiliki 6 *node* dengan 7 *link* dimana sebagai contoh pada kasus ini akan dimulai dari *Node A* menuju *Node F*. Tahapan untuk menyelesaikan permasalahan ini kemudian dijabarkan pada bagian di bawah ini. Sementara untuk memudahkan proses perhitungan, akan digunakan *template* label seperti yang terlihat pada Gambar 12.2.

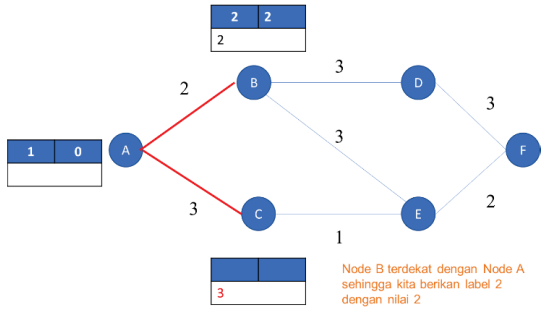


Gambar 12.2 Template label.

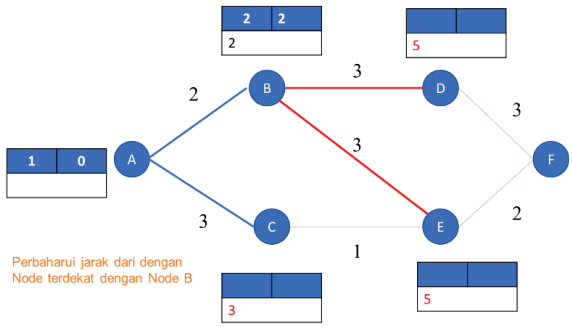


Gambar 12.3 Pembaharuan *working value Node* terdekat.

Tahap pertama dalam proses perhitungan ini adalah menganggap semua *node* selain *node* asal sebagai ***node* non permanen**. Setelah itu, anggap *node* asal sebagai *node* aktif, dimana *node* terdekat adalah *node* B dan *node* C. Kemudian hitung jarak kumulatif dari *node* asal (A) menuju *node* terdekat (i.e., B dan C). Terlihat pada **Gambar 12.3** *working value* pada *Node* B dan *Node* C berubah sesuai dengan jaraknya dari *node* asal. Dari kedua *node* ini, terlihat bahwa *Node* B memiliki nilai *working value* terkecil sehingga *node* B dapat dilabeli sebagai ***Node* Permanen (lihat Gambar 12.4)**.

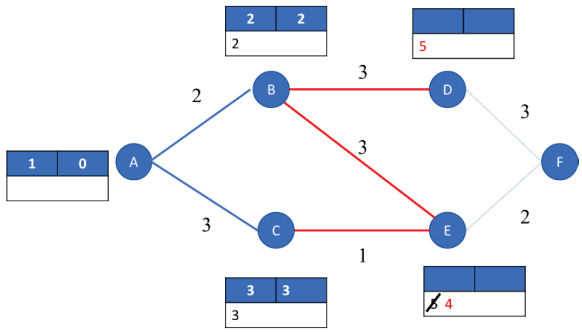


Gambar 12.4 Pembaharuan Label *Node* terdekat.



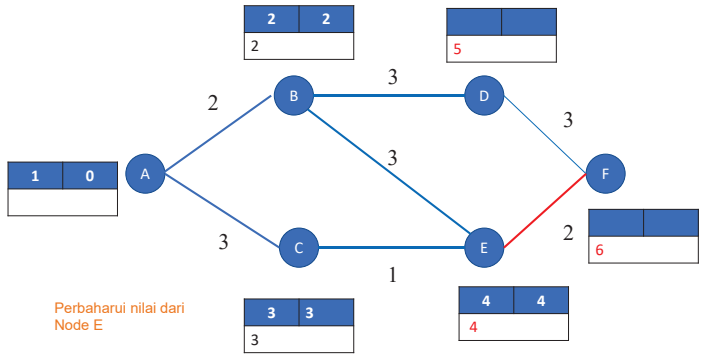
Gambar 12.5 Pembaharuan Label *Node* pada tahap kedua.

Tahapan selanjutnya pada prinsipnya mengulangi proses terkait pembaharuan label pada ***node* non permanen. *Node* permanen B** kemudian dianggap sebagai *node* aktif dan dilakukan evaluasi pada *node* terdekat yaitu *Node* D dan E. Gambar 12.5 menunjukkan adanya perubahan *working value* pada *Node* D dan E. Kemudian dari *working value* yang ada dapat terlihat bahwa *Node* C memiliki *working value* terkecil sehingga dapat dilabeli sebagai label permanen.



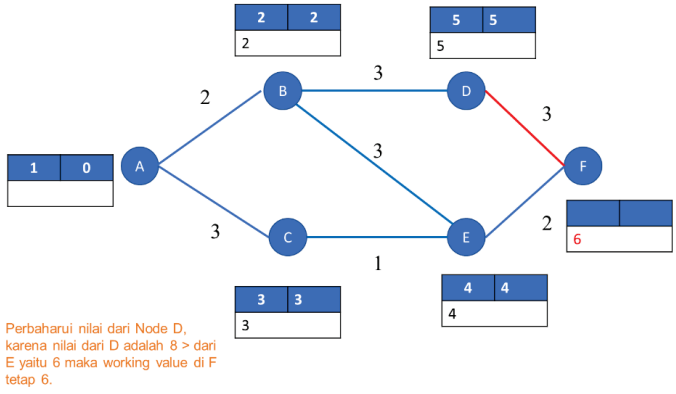
Gambar 12.6 Pembaharuan Label *Node* pada tahap ketiga.

Karena *node* permanen adalah *Node* C maka akan dievaluasi *node* terdekat dari *Node* C ini, seperti yang terlihat pada Gambar 12.6 Selain itu, pada Gambar 12.6 terjadi pembaharuan *working value* pada *Node* E karena kumulatif jarak dari A menuju E melalui *Node* C lebih kecil dibandingkan jika melalui *Node* B. Dari Gambar 12.6 terlihat bahwa nilai *working value* terkecil saat ini adalah *Node* E sehingga kita jadikan *Node* E sebagai ***Node* Permanen**. *Node* terdekat dari *Node* E kemudian dievaluasi (i.e., *Node* F) seperti yang terlihat pada Gambar 12.7.



Gambar 12.7 Pembaharuan Label *Node* pada tahap empat.

Proses pembaharuan label dilakukan selanjutnya dengan melihat *working value* yang tersisa dimana *Node* D memiliki *working value* terkecil. *Node* D dijadikan ***node* permanen** dengan *node* terdekat adalah *node* F. Nilai *working value* pada *Node* F tidak diperbaharui karena jarak kumulatif dari A menuju F melalui D masih lebih besar dibandingkan dari *Node* E. Terakhir jadikan *Node* F sebagai *Node* Permanen, dan proses ini *Djikstra* selesai.

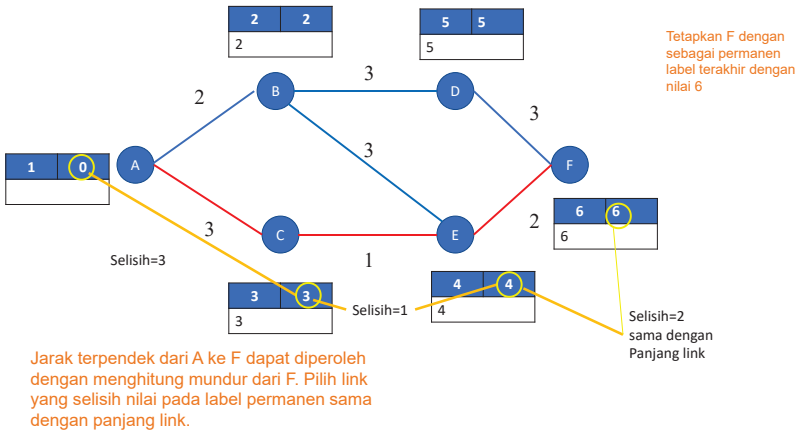


Gambar 12.8 Pembaharuan Label *Node* pada tahap lima.

Tahap terakhir dari metode ini adalah menghitung secara *backward* hasil dari proses yang dilakukan, yang diperoleh jalur terpendeknya adalah A-C-E-F. Proses perhitungan dimulai dari *node* tujuan, yaitu *Node* F, dimana F memiliki dua buah *node* terdekat yaitu *Node* E dan *Node* D. Untuk menentukan *node* yang akan dipilih dapat diterapkan aturan sederhana berikut:

"*Node* yang termasuk jalur terpendek merupakan *node* yang memiliki selisih nilai label permanen yang sama besar dengan jarak antar *node*nya, (i.e., panjang *link*nya)."

Sebagai contoh, lihat Gambar 12.8, dimana jalur F→E dipilih karena selisih nilai labelnya adalah 2 yang berkesesuaian dengan panjang *link* E-F. *Link* D-F tidak dapat dipilih karena, selisih antar labelnya adalah 1 (i.e., 6 dikurang 5) sementara jarak antar D-F adalah 3. Sehingga untuk memperoleh jalur terpendek *node* F akan menuju ke *node* E. Proses ini dilakukan terus menerus hingga *node* tujuan (i.e., *node* F) terhubung dengan *node* akhir (i.e., *node* A).



Gambar 12.9 Hasil akhir *Djikstra*

12.2. Lagrange Relaxation

Metode lain yang umum ditemui untuk menyelesaikan permasalahan optimasi kombinatorial di bidang transportasi adalah, *Lagrange Relaxation*. Metode *lagrange* umumnya digunakan untuk menyelesaikan *lower bound* dari permasalahan optimasi. Jika mengibaratkan pencarian solusi optimal seperti mencari titik yang memberikan hasil paling maksimal/minimal pada sebuah garis (lihat Gambar 12.10), maka solusi optimal pasti akan berada di antara di batas atas (i.e., *upper bound*) dan batas bawah (i.e., *lower bound*). Untuk mempermudah pencarian, batas atas dan batas bawah ini harus sedemikian

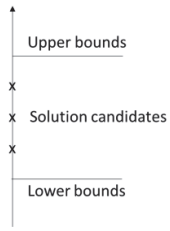
rupa berjarak sangat dekat dengan solusi optimal. Artinya proses pencarian solusi optimum akan berada pada rentang terbatas saja. Dalam praktiknya, *upper bound* diperoleh dengan menerapkan metode *heuristik* (e.g., *greedy*) maupun *metaheuristik*, yang umumnya menghasilkan solusi awal yang dianggap sebagai batas atas permasalahan optimasi.

Sementara salah satu teknik solusi yang umum digunakan untuk menemukan batas bawah adalah *relaksasi lagrange*. Metode ini menjadi salah satu primadona karena beberapa fakta berikut:

- Permasalahan optimasi kombinatorial terkadang hanya merupakan bagian *easy optimization problem* (i.e., *polynomial time problem*), akan tetapi karena adanya kompleksitas dari fungsi kendala menyebabkan permasalahan optimasi berubah menjadi *hard problem* (i.e., *non polynomial time problem*).
- Proses relaksasi fungsi kendala kepada fungsi tujuan dapat dilakukan, yang dapat menyederhanakan permasalahan optimasi sehingga lebih mudah ditangani.
- Pada banyak aplikasi, penggunaan metode *lagrange relaxation* memiliki waktu komputasi yang masuk akal sehingga menjadi pilihan untuk menyelesaikan permasalahan yang kompleks.

Metode ini pada prinsipnya melibatkan beberapa tahap berikut:

i) formulasi permasalahan dalam bentuk *integer programming*
ii) relaksasi fungsi kendala ke dalam fungsi tujuan
iii) selesaikan permasalahan optimasi dalam bentuk integer yang telah direlaksasi, proses penyelesaiannya dapat menggunakan metode *subgradien* ataupun *multiplier.*



Gambar 12.10 Ilustrasi upper bounds and lower bounds permasalahan optimasi.

Dalam bidang transportasi, metode ini diaplikasikan untuk menyelesaikan banyak kasus, diantara permasalahan *travelling salesman problem* dan permasalahan optimasi

pemeliharaan jaringan jalan. Meskipun dalam buku ini, penerapan *lagrange relaxation* akan dibahas dengan mengkonsiderasikan permasalahan optimasi pemeliharaan jalan.

Contoh penerapan metode *Lagrange Relaxation*

Anggaplah terdapat dua buah segmen jalan yang akan dipelihara dengan dua jenis tipe pemeliharaan yaitu pemeliharaan rutin dan pelapisan ulang. Informasi awal dari kedua segmen adalah sebagai berikut:

Tabel 12.3 Contoh segmen jalan untuk aplikasi *lagrange relaxation.*

Nama Segmen	Panjang (m)	Lebar (m)	Nilai IRI awal	Prediksi kenaikan IRI per tahun
A	200	3	6	1
B	200	3	3	2

Sementara diketahui pula bahwa pemeliharaan rutin akan menurunkan nilai IRI sebesar 0,5 sementara pelapisan ulang akan menyebabkan nilai IRI menjadi 2. Jika diketahui bahwa pemeliharaan rutin membutuhkan biaya sebesar Rp. 0,5 juta/m^2 sementara pelapisan ulang membutuhkan 2 juta/m^2. *Budget* yang tersedia setiap tahunnya adalah Rp. 2.000 juta atau sekitar Rp. 3.3 juta m^2 untuk lebar dan panjang tersebut. Berikan rekomendasi penanganan untuk kedua ruas tersebut.

Tahap pertama adalah memformulasikan permasalahan tersebut dalam bentuk *integer programming*.

$$\min\ f(x) = \sum_{t \in T} \sum_{j \in J} \sum_{i \in I} y^i_{(t,j)} x^i_j \tag{12.1}$$

Fungsi tujuan dari permasalahan optimasi ini anggaplah sebagai meminimalkan IRI $y^i_{(t,j)}$ pada jaringan jalan pada waktu *t*. Dimana *i* menunjukkan indeks dari jalan, sementara *j* adalah indeks dari aksi pemeliharaan yang dapat dilakukan (i.e., 1= tidak dipelihara, 2=pemeliharaan rutin, 3= pelapisan ulang) adalah variable keputusan yang bernilai biner, dimana $x^i_j = 1$ berarti adalah pemeliharaan-*j* akan diaplikasikan, sementara $x^i_j = 0$ berarti sebaliknya.

Setelah dapat didefinisikan fungsi tujuannya, maka tahapan selanjutnya adalah mendefinisikan fungsi kendala, yaitu adanya batasan dari sisi anggaran pertahunnya. Karena segmen tersebut memiliki lebar dan panjang yang sama, maka dapat variabel biaya pemeliharaannya dapat hanya berupa unit biaya pada masing-masing jenis pemeliharaannya saja. Dimana total biaya untuk mengimplementasikan keputusan pemeliharaan harus kurang atau sama dengan *budget* yang tersedia pada setiap tahunnya.

$$\sum_{i \in I} \sum_{j \in J} c^j x_i^j \leqslant B_t \tag{12.2}$$

Hal lain yang penting untuk dipastikan adalah bahwa setiap segmen hanya dapat ditangani oleh satu jenis pemeliharaan saja, sehingga fungsi kendala berikut ditambahkan.

$$\sum_{j \in J} \sum_{i \in I} x_i^j = 1 \tag{12.3}$$

Setelah permasalahan telah diformulasikan dalam bentuk *integer programming*, maka tahapan selanjutnya adalah merelaksasi fungsi kendala masuk ke dalam fungsi tujuan. Relaksasi ini menyebabkan adanya tambahan pada fungsi tujuan (i.e., fungsi kendala) yang dikalikan dengan variabel *lagrange*. Pada kasus pemeliharaan jalan, fungsi kendala yang direlaksasi adalah batasan biaya yang berperan membuat kompleksitas dari permasalahan optimasi ini. Sehingga dengan merelaksasi fungsi kendala tersebut, diharapkan proses penyelesaiannya akan jauh lebih sederhana. Hasil relaksasi fungsi kendala biaya dapat dilihat pada formulasi dibawah ini.

$$\min\ f(x) = \sum_{t \in T} \left[\sum_{j \in J} \sum_{i \in I} y_{(t,j)}^i x_j^i + \gamma_{(t,1)} \left(\sum_{i \in I} \sum_{j \in J} c^j x_i^j - B_t \right) \right] \tag{12.4}$$

$s.t$

$$\sum_{j \in J} \sum_{i \in I} x_i^j = 1 \tag{12.5}$$

$$x_i^j \in \{0, 1\}$$

$\gamma_{(t,1)}, \gamma_{(t,2)}, \gamma_{(t,3)}$ merupakan *lagrange multiplier* yang berkaitan dengan relaksasi fungsi kendala biaya. Dengan mendasarkan kepada relaksasi tersebut maka permasalahan pada contoh kasus dapat dituliskan pada tahun *t* sebagai berikut.

$$\begin{aligned} \min\ f(x) = {} & (y_1^1 x_1^1 + y_1^2 x_1^2 + y_1^3 x_1^3 + y_2^1 x_2^1 + y_2^2 x_2^2 + y_2^3 x_2^3) \\ & + \gamma_{(t,1)} \left((c^1 (x_1^1 + x_2^1) + c^2 (x_1^2 + x_2^2) + c^3 (x_1^3 + x_2^3)) - B_t \right) \end{aligned} \tag{12.6}$$

Persamaan di atas dapat disederhanakan dengan melakukan re-*grouping* berdasarkan nilai *x*-nya.

$$\begin{aligned} & f(x) = \\ & [\ x_1^1 (y_1^1 + \gamma_{(t,1)} c^1) + x_1^2 (y_1^2 + \gamma_{(t,1)} c^2) + x_1^3 (y_1^3 + \gamma_{(t,1)} c^3) \\ & + x_2^1 (y_2^1 + \gamma_{(t,1)} c^1) + x_2^2 (y_2^2 + \gamma_{(t,1)} c^2) + x_2^3 (y_2^3 + \gamma_{(t,1)} c^3) - \gamma_{(t,1)} B_t] \end{aligned} \tag{12.7}$$

$$
\begin{aligned}
&karena\ c^1 = 0 \\
&f(x) = \\
&[\ x_1^1(y_1^1) + x_1^2(y_1^2 + \gamma_{(t,1)}c^2) + x_1^3(y_1^3 + \ \gamma_{(t,1)}c^3) \\
&+x_2^1(y_2^1) + x_2^2(y_2^2 + \gamma_{(t,1)}c^2) + x_2^3(y_2^3 + \gamma_{(t,1)}c^3) - \gamma_{(t,1)}B_t]
\end{aligned}
\tag{12.8}
$$

Nilai y_i^j, c^j, B_t dapat dihitung berdasarkan informasi di atas sebagai contoh pada segmen A (i.e., *i*=1) tanpa pemeliharaan (i.e., *j*=1), maka nilai IRInya adalah $y_1^1 = 6 + 1 = 7$. Hal ini disebabkan tanpa adanya pemeliharaan nilai IRI pada segmen A akan naik, dari sebelumnya bernilai IRI =6 menjadi bernilai IRI=7. Nilai IRI untuk setiap keputusan pada masing-masing segmen diresumekan oleh Tabel 12.4.

Tabel 12.4 Nilai IRI pada masing-masing keputusan.

	IRI jika		
	j=1	*j*=2	*j*=3
Segmen A (*i*=1)	7,0	6,5	2,0
Segmen B (*i*=2)	5,0	4,5	2,0

Atas dasar tersebut maka fungsi tujuan dapat disederhanakan menjadi

$$
\begin{aligned}
&f(x) = \\
&[\ 7x_1^1 + x_1^2(6,5 + 0,5\gamma_{(t,1)}) + x_1^3(2 + 2\ \gamma_{(t,1)}) \\
&+5x_2^1 + x_2^2(4,5 + 0,5\gamma_{(t,1)}) + x_2^3(2 + 2\gamma_{(t,1)}) - 3,3\gamma_{(t,1)}]
\end{aligned}
\tag{12.9}
$$

$$
\begin{aligned}
&Jika \\
&C_1 = (6,5 + 0,5\gamma_{(t,1)}) \\
&C_2 = (2 + 2\gamma_{(t,1)}) \\
&C_3 = (4,5 + 0,5\gamma_{(t,1)}) \\
&maka \\
&\min\ f(x) = \\
&[\ 7x_1^1 + C_1x_1^2 + C_2x_1^3 + 5x_2^1 + C_3x_2^2 + C_2x_2^3 - \ 3,3\gamma_{(t,1)}]
\end{aligned}
\tag{12.10}
$$

Nilai yang dihasilkan dari *f*(*x*) dapat dianggap sebagai nilai batas bawah (i.e., Z_{LB}) dari permasalahan optimasi yang berdekatan dengan solusi optimal. Untuk menyelesaikan permasalahan ini dapat diawali dengan mencoba menentukan nilai awal dari *multiplier* ($\gamma_{(t,1)}, \gamma_{(t,2)}, \gamma_{(t,3)}$). Misal anggap $\gamma_{(t,1)} = 1,1$ maka

$$
\begin{aligned}
&C_1 = (6.5 + 1.1(0.5)) = 7.05 \\
&C_2 = (2 + \ 1.1(2)) = 4.2 \\
&C_3 = (4.5 + 1.1(0.5)) = 5.05 \\
&[\ 7x_1^1 + C_1x_1^2 + C_2x_1^3 + 5x_2^1 + C_3x_2^2 + C_2x_2^3 - \ 3,3\gamma_{(t,1)}]
\end{aligned}
\tag{12.11}
$$

$$\begin{aligned}&\min Z_{LB}\\&=7x_1^1+7,05x_1^2+4,2x_1^3+5x_2^1+5,05x_2^2+4,2x_2^3-3,3(1,1)\\&=7x_1^1+7,05x_1^2+4,2x_1^3+5x_2^1+5,05x_2^2+4,2x_2^3-3.63\end{aligned} \quad (12.12)$$

Dengan mendasarkan kepada nilai *multiplier* yang ditentukan sebelumnya, diketahui dapat dengan mudah diketahui konfigurasi *x* yang membuat nilai solusi batas bawah minimal adalah kita menetapkan $x_1^3=1;x_2^3=1;$ karena memiliki nilai koefisien terkecil yang menyebabkan nilai Z_{LB} menjadi minimum.

$$\begin{aligned}&x_1^1=0;x_1^2=0;x_1^3=1;x_2^1=0;x_2^2=0;x_2^3=1;\\&Z_{LB}=7(0)+7,05(0)+4,2(1)+5(0)+5,05(0)+4,2(1)-3,63\\&Z_{LB}=4,77\end{aligned} \quad (12.13)$$

Meskipun untuk memperoleh hasil optimal, nilai *lagrange multiplier* harus ditentukan secara tepat, yang umumnya ditentukan dengan penggunaan metode *subgradient* ataupun *multiplier adjustment*. Kedua metode ini pada prinsipnya berusaha untuk menentukan nilai *multiplier* secara terstruktur dan menetapkan kriteria berhentinya secara konsisten. Pada buku ini akan disampaikan salah satu contoh prosedur *multiplier adjustment* yang sederhana, dengan tahapan sebagai berikut:

1. Inisialiasi nilai *multiplier* dan tetapkan total iterasi dalam perhitungan. Serta tetapkan *iter*=1
2. Tentukan nilai Z_{LB} berdasarkan nilai *multiplier*
3. Hitung *subgradient* (i.e., G_j) dari fungsi kendala yang direlaksasi
4. Tentukan nilai *multiplier* selanjutnya dengan mendasarkan kepada fungsi berikut:

$$\gamma_{(t,j)}=\max(0,\gamma_{(t,j)}+TG_j)\ ,\ \text{dimana}\ T\text{: }step\ size$$

Jika diketahui bahwa *T* adalah *step size* yang bersifat tetap (dalam metode *subgradient* nilai ini berubah didasarkan kepada suatu fungsi tertentu), maka dari persamaan di atas dapat disimpulkan bahwa:

- Nilai *multiplier* **akan bertambah jika** $G_j>0$
- Nilai *multiplier* **akan tetap** jika $G_j=0$
- Nilai *multiplier* **akan berkurang** jika $G_j<0$

5. Perhitungan dihentikan jika jumlah iterasi yang telah dilakukan melewati total iterasi yang ditetapkan sebelumnya.

Sebagai contoh, tahapan diatas akan coba diimplementasikan pada studi kasus terkait pemeliharaan jalan, sebagai berikut:

1. Inisialiasi nilai *multiplier,* tetapkan (misal) total iterasi=4, *iter*=1
2. Dipilih $\gamma_{(t,1)} = 1,1$ sehingga diperoleh $x_1^3 = 1; x_2^3 = 1;$
3. Tentukan nilai Z_{LB} berdasarkan nilai *multiplier*
4. Setelah dihitung diperoleh $Z_{LB} = 4,77$
5. Hitung *subgradient* (i.e., G_j) dari fungsi kendala yang direlaksasi, dimana dalam kasus ini adalah:

$$G_1 = \left(\sum_{i \in I} \sum_{j \in J} c^j x_i^j - B_t \right) \qquad (12.14)$$
$$G_1 = ((0(0+0) + 0.5(0+0) + 2(1+1)) - 3.3) = 0.7$$

6. Tentukan nilai *multiplier* selanjutnya dengan mendasarkan kepada fungsi berikut, anggap *step size* adalah 0,2:

$$\gamma_{(t,j)} = \max(0, \gamma_{(t,j)} + TG_j) \text{ , } dimana\ T = 0,2$$
$$\gamma_{(t,1)} = \max(0, 1, 1 + (0,2)(0,7)) \qquad (12.15)$$
$$\gamma_{(t,1)} = \max(0, 1.24) = 1,24$$

7. *iter*=*iter*+1, lanjutkan perhitungan ke langkah 2 hingga iter>total iterasi

Terlihat dari contoh di atas, bahwa solusi yang direkomendasikan pada iterasi pertama masih melewati batasan biaya, sehingga *subgradien* masih bernilai positif. Karena bernilai positif, maka dengan formula di atas, nilai *multiplier* akan bertambah kemudian nilai C_2 akan naik pula sehingga konfigurasi kandidat solusi (x_i^j) akan berubah pula. Adapun resume perhitungannya dapat dilihat pada Tabel 12.5 di bawah ini.

Tabel 12.5 Implementasi *multiplier adjustment.*

Iterasi ke-	$\gamma_{(t,1)}$	C	x_i^j	Z_{LB}	G_j
1	1,1	$C_1 = 7,05$ $C_2 = 4,2$ $C_3 = 5,05$	$x_1^1 = 0; x_1^2 = 0; x_1^3 = 1;$ $x_2^1 = 0; x_2^2 = 0; x_2^3 = 1;$	4,77	0,7
2	1,24	$C_1 = 7,12$ $C_2 = 4,48$ $C_3 = 5,15$	$x_1^1 = 0; x_1^2 = 0; x_1^3 = 1;$ $x_2^1 = 0; x_2^2 = 0; x_2^3 = 1;$	4,87	0,7
3	1,38	$C_1 = 7,19$ $C_2 = 4,76$ $C_3 = 5,22$	$x_1^1 = 0; x_1^2 = 0; x_1^3 = 1;$ $x_2^1 = 0; x_2^2 = 0; x_2^3 = 1;$	4,96	0,7
4	1,52	$C_1 = 7,26$ $C_2 = 5,04$ $C_3 = 5,29$	$x_1^1 = 0; x_1^2 = 0; x_1^3 = 1;$ $x_2^1 = 1; x_2^2 = 0; x_2^3 = 0;$	4,18	-0,8

Berdasarkan tabel di atas, dapat diambil kesimpulan bahwa solusi yang ditawarkan oleh *lagrange relaxation* dengan *multiplier adjustment* adalah menetapkan segmen A dilapis ulang, sementara segmen B dibiarkan saja pada tahun pertama. Untuk tahun selanjutnya, dilakukan proses yang sama. Solusi yang dihasilkan oleh *multiplier adjustment* pada prinsipnya sangat tergantung pada *setting step size* dan jumlah iterasinya yang terkadang menghasilkan solusi yang lokal optimal. Oleh karenanya, telah berkembang begitu banyak metode untuk menyelesaikan permasalahan *lagrange relaxation* dengan hasil yang lebih baik yang tidak dibahas di dalam buku ini.

Pembaca yang tertarik untuk melihat penjelasan dalam bentuk audio-visual terkait Bab ini dapat mengunjungi playlist youtube berikut:

https://bit.ly/Playlist-ShortestPath

13. Optimasi Pengembangan Jaringan Transportasi dengan OPTANT PL

Bagian ini membahas permasalahan optimasi dalam kerangka pengembangan jaringan transportasi. Secara spesifik permasalahan ini akan diselesaikan dengan bantuan perangkat lunak OPTANT PL. Nama ini merupakan akronim dari optimasi jaringan transportasi (i.e., OPTANT) dengan pembebanan lalu lintas (i.e., PL). Perangkat lunak ini dikembangkan untuk mampu melakukan perhitungan sebaran pergerakan, pembebanan lalu lintas hingga menyelesaikan permasalah optimasi dalam pengembangan jaringan transportasi. Perangkat lunak ini dikembangkan oleh peneliti dari Kelompok Keahlian Rekayasa Transportasi, Fakultas Teknik Sipil dan Lingkungan, Institut Teknologi Bandung.

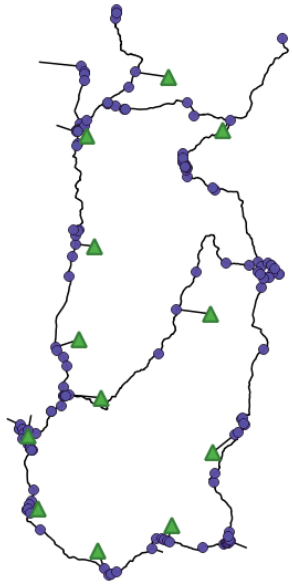
13.1. Studi Kasus Pengembangan Jaringan Transportasi

Pengembangan jaringan transportasi pada dasarnya dapat berupa:

- pembangunan ruas baru (yaitu, pembangunan jalan, jalur kereta api, jalur kapal laut, jalur pesawat),
- peningkatan kapasitas ruas eksisting (yaitu, pelebaran jalan, double track KA dsb),
- pembangunan simpul/terminal baru (yaitu, pembangunan bandara, terminal bus, terminal kapal, stasiun KA baru),
- peningkatan kapasitas simpul/terminal eksisting (yaitu penambahan terminal, penambahan alat dsb)

Meskipun dalam OPTANT PL ver 1.2 ini masih dibatasi pada jenis pengembangan yang kedua yaitu dalam rangka peningkatan kapasitas eksisting. Sebagai studi kasus pada buku ini, akan dibahas jaringan jalan sederhana dengan jumlah segmen 176 segmen yang disusun oleh 164 *node* (lihat Gambar 13.1). Jaringan sederhana ini memiliki 12 *centroid* yang menjadi asal dan tujuan dari pergerakan.

*) OPTANT PL tersedia untuk kebutuhan akademik, permintaan dapat disampaikan melalui alamat surat elektronik berikut: *febri.zukhruf@ftsl.itb.ac.id*



Gambar 13.1 Jaringan Jalan Studi Kasus.

Adapun kondisi sebelum dilakukan pengembangan dapat dilihat pada Gambar 13.2, dimana terdapat beberapa segmen yang memiliki VCR lebih dari 1.

Gambar 13.2 *Vehicle Capacity Ratio* (VCR) hasil luaran OPTANT PL.

Untuk pengembangan jaringan jalan, anggaplah kita hanya akan terfokus kepada ruas dengan VCR tinggi hingga sedang. Sehingga pada contoh kasus ini dipilih 8 segmen yang akan dtingkatkan kapasitasnya, seperti yang terlihat pada Tabel 13.1 dibawah ini.

Tabel 13.1 Pilihan segmen untuk pelebaran jalan.

No	Nama Segmen	Lebar Eksisting	VCR Esksiting	Kapasitas Setelah dilebarkan	Biaya Pelebaran
1	BTS. KOTA SUNGGUMINASA BTS. KAB. TAKALAR	8,2	1,1	5214,5	47,4
2	BTS. KAB. GOWA BTS. KOTA TAKALAR	6,2	1,5	4250,6	12,3
3	JLN. DIPONEGORO (TAKALAR)	7,9	1,1	5081,1	4,2
4	JLN. SUDIRMAN (TAKALAR)	8,5	0,8	5392,9	5,7
5	BTS. KOTA TAKALAR BTS. KAB. TAKALAR/BTS. KAB. JENEPONTO	6,1	1,1	4201,7	27,4
6	BTS. KAB. TAKALAR/BTS. KAB. JENEPONTO BTS. KOTA JENEPONTO	6,1	1,1	4201,7	85,7
7	IMPA IMPA - TARUMPAKKAE	6,3	0,6	4319,0	70,5
8	BAJO - ARASOE (KM. 260)	5,8	0,7	4042,0	87,5

Atas dasar informasi di atas, permasalahan optimasi dapat disusun sebagai berikut:

- Fungsi tujuan: meminimalkan total waktu perjalanan di jaringan, memaksimalkan selisih total waktu perjalanan sebelum dan sesudah dilakukan pelebaran,
- Kendala: biaya
- Variabel keputusan: segmen yang dipilih untuk dilebarkan.

OPTANT PL versi 1.2 dapat menyelesaikan permaslahan optimasi seperti di atas dengan melepaskan fungsi kendala terkait biaya menjadi bagian dari fungsi tujuan. Sehingga, fungsi tujuannya adalah memaksimalkan rasio dari selisih total waktu perjalanan sebelum dan sesudah dilakukan pelebaran dengan biaya untuk pelebaran. Secara sederhana permasalahan optimasi ini dapat dituliskan sebagai berikut:

$$\max f(x) = \frac{\sum_{a \in A} t_a^0 y_a^0 - \sum_{a \in A} t_a(x) y_a(x)}{c(x)} \tag{13.1}$$

Dimana:

$f(x)$: fungsi tujuan permasalahan optimasi

t_a^0: waktu perjalanan pada *link a* pada kondisi tanpa adanya keputusan pelebaran

y_a^0: arus yang melewati *link a* pada kondisi tanpa adanya keputusan pelebaran

$t_a(x)$: waktu perjalanan pada *link a* dengan diimplementasikannya keputusan pelebaran-*x*

$y_a(x)$: arus yang melewati *link a* dengan diimplementasikannya keputusan pelebaran-*x*

$c(x)$: biaya implementasi keputusan pelebaran-*x*

13.2. Tahapan Implementasi dengan OPTANT PL

Bagian ini membahas prosedur melakukan optimasi pada pengembangan jaringan transportasi. Untuk menggunakan fitur optimasi, pengguna harus memastikan bahwa *database* yang telah disiapkan dalam *template spreadsheet* terisi secara sempurna. Meskipun pada bagian ini hanya akan dibahas terkait *Database* Pengembangan Jaringan Transportasi (**datopt**).

13.2.1. Database Pengembangan Jaringan Transportasi

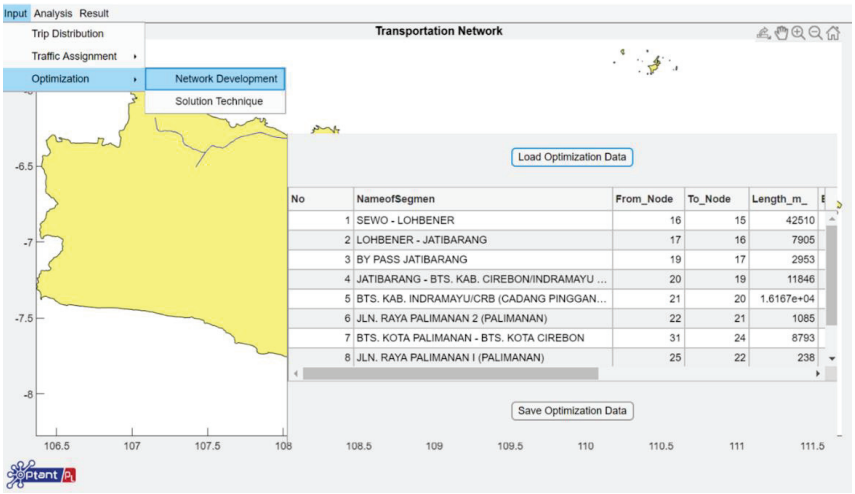
Tahapan pertama adalah menyusun *database* pengembangan jaringannya, dengan mengunakan *template spreadsheet* pada tab **datopt.** Template tersebut memiliki delapan informasi utama, yang harus diisi sesuai dengan kolomnya. Sebelum melakukan pengisian, sebaiknya dipilih terlebih dahulu ruas yang akan dijadikan pilihan untuk dikembangkan, ruas yang dipilih harus dilengkapi dengan nama dan *node* awal dan akhirnya. Setelah itu, rancangan pengembangan berupa peningkatan kapasitas ditentukan melalui pengisian estimasi peningkatan kapasitas. Selain itu, informasi terkait biaya yang dibutuhkan untuk mengembangkan alternatif pilihan tersebut disampaikan pada bagian terakhir (lihat Tabel 13.2).

Tabel 13.2 Keterangan data dalam datopt.

No	Nama data	Keterangan	Jenis Data
1	Name of Segmen	nama ruas	teks
2	From_*Node*	nomor *node* awal dari segmen, sesuai penamaan dalam peta (jika ada)	numerik
3	To_*Node*	nomor *node* akhir dari segmen, sesuai penamaan dalam peta (jika ada)	numerik
4	*Length* (m)	panjang segmen dalam satuan meter	numerik
5	*Existing Width*	Lebar segmen	numerik
6	*Additional Width*	Penambahan lebar	numerik
7	*Expected New Capacity*	Perkiraan penambahan kapasitas	numerik
8	*Estimated Cost*	Perkiraan biaya investasi yang dibutuhkan	numerik

13.2.2.Input *Database* Pengembangan Jaringan Transportasi

Proses memasukkan database kedalam OPTANT PL dilakukan dengan menggunakan menu ***input > optimization > Network Development.*** Aplikasi akan mengarahkan pengguna untuk memilih lokasi dokumen database yang telah disiapkan pada *template spread sheet*. Jika hasil dari ***Load Optimization Data*** sesuai dengan yang terdapat di data database maka tekan ***Save Optimization Data***.

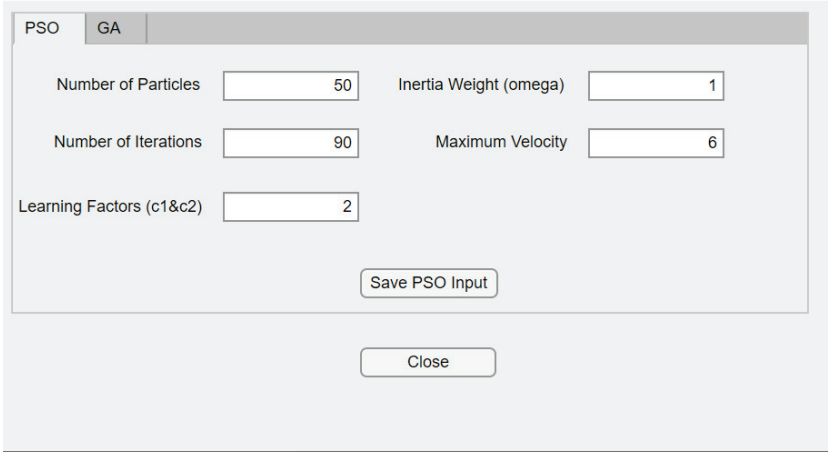


Gambar 13.3 Tangkapanlayar Input Load Optimization Data.

13.2.3. Input Parameter Metode Penyelesaian Masalah Optimasi

Penyelesaian masalah optimasi dilakukan dengan menerapkan metode SGA ataupun DBPSO. Kedua metode ini membutuhkan beberapa *input* untuk menyelesaikan permasalahan optimasi. Proses memasukkan *input* dapat dilakukan melalui menu ***input > optimization > solution technique***. DBPSO membutuhkan beberapa input yaitu:

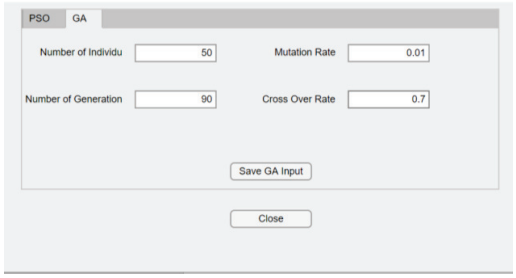
- ***Number of particles,*** jumlah partikel dalam satu *swarm* dengan nilai minimum 2
- ***Number of iterations,*** jumlah iterasi dengan nilai minimum 1
- ***Learning Factors,*** faktor belajar dengan nilai antara 1 s.d 4
- ***Inertia Weight,*** faktor bobot dengan nilai antara 0.8 s.d 1.8
- ***Maximum Velocity***, kecepatan maksimum partikel PSO dengan nilai minimum 1



Gambar 13.4 Tangkapan layar *Input* PSO.

Sementara untuk parameter input GA adalah:

- ***Number of Individu***, jumlah individu dalam satu generasi
- ***Number of Generation***, jumlah generasi
- ***Mutation Rate*,** rate terjadinya mutasi dengan nilai 0.01 hingga 1
- ***Cross Over Rate*,** rate terjadinya kawin silang dengan nilai 0.01 hingga 1

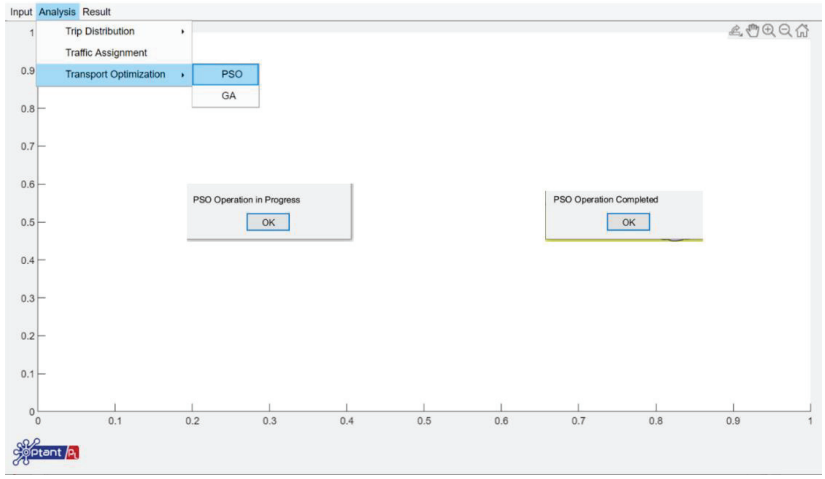


Gambar 13.5 Tangkapan layar Input PSO.

Setelah semua input dimasukkan, tekan tombol **Save** dan **Close**.

13.2.4. Penyelesaian Permasalahan Optimasi

Tahapan selanjutnya dalam menyelesaikan permasalahan optimasi pengembangan jaringan transportasi adalah memanggil DBPSO dan SGA untuk mencari solusi terbaik. Proses ini dilakukan dengan menekan menu ***Analysis > Transport Optimization > PSO*** dan ***Analysis > Transport Optimization > GA***. Proses dimulainya analisis ditandai dengan adanya tanda ***PSO Operation in Progress*** pada PSO serta diakhir dengan adanya tanda ***PSO Operation Completed***.



Gambar 13.6 Tangkapan layar penyelesaian masalah optimasi dengan PSO.

13.2.5. Luaran OPTANT PL

Hasil utama dari fitur optimasi adalah memberikan solusi optimal dari permasalahan optimasi yang didefinisikan. Solusi ini diberikan dengan menuliskan apakah alternatif solusi atau keputusan akan diambil atau tidak (lihat gambar dibawah ini).

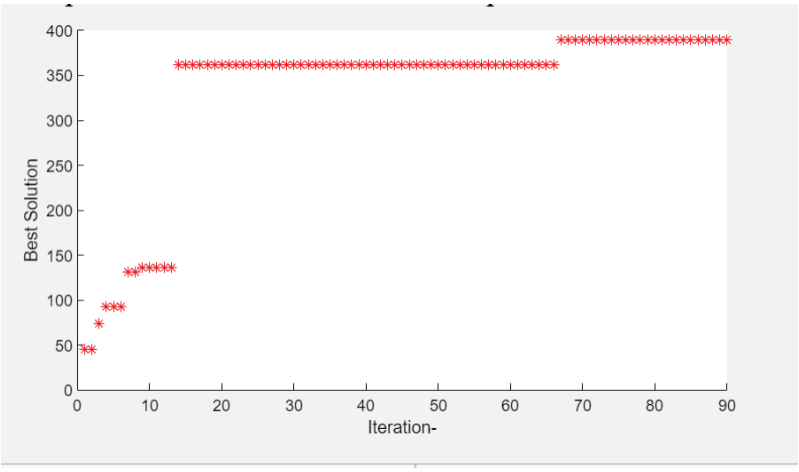
Selain informasi terkait keputusan optimal, terdapat pula beberapa parameter yang disediakan ole OPTANT PL yaitu:

- ***Computation time***, waktu komputasi yang dibutuhkan untuk menyelesaikan permasalahan optimasi.
- ***Optimal solution value,*** *fitness value* maksimum yang merupakan pembagian *benefit* dibandingkan dengan biaya.
- ***Expected benefit,*** *benefit* yang diperkirakan diperoleh dengan diimplementasikannya solusi optimal hasil perhitungan.
- ***Estimated action cost***, *benefit* yang diperlukan untuk diimplementasikannya solusi optimal hasil perhitungan.

From Node	To Node	Name of Segmen	Decision
1	2	JLN. URIP SUMOHARJO (CIKARANG)	Not Selected
2	3	BTS. KOTA CIKARANG - BTS. KOTA KARAWANG	Not Selected
3	4	JLN. PANGKAL PERJUANGAN (KARAWANG)	Not Selected
4	5	LINGKAR KARAWANG	Not Selected
5	6	BTS. KOTA KARAWANG - BTS. KOTA CIKAMPEK	Not Selected
6	7	JLN. RAYA DAWUAN (CIKAMPEK)	Not Selected
7	8	JLN. JEND. A. YANI (CIKAMPEK)	Not Selected
9	10	BTS. KOTA CIKAMPEK - BTS. KAB. SUBANG/ KAR…	Not Selected
8	11	JLN. JEND. SUDIRMAN (CIKAMPEK)	Not Selected
11	9	JLN. RAYA JATISARI (CIKAMPEK)	Not Selected
10	12	BTS. KAB. SUBANG/KARAWANG - BTS. KOTA PA…	Not Selected
12	13	JLN. EYANG TIRTAYASA (PAMANUKAN)	Selected
14	15	BTS. KOTA PAMANUKAN - SEWO	Not Selected
13	14	JLN. H. SYAHBANA (PAMANUKAN)	Not Selected
16	15	SEWO - LOHBENER	Not Selected
17	16	LOHBENER - JATIBARANG	Not Selected

Gambar 13.7 Tangkapan layar keputusan optimal dengan PSO.

Hal lain yang tidak kalah penting adalah mengukur kinerja dari metode optimasinya, dalam hal ini dapat direpresentasikan dengan waktu memperoleh solusi terbaik. Semakin cepat memperoleh solusi optimal (yaitu dalam iterasi rendah) maka dapat dikatakan metode tersebut merupakan metode yang efisien.

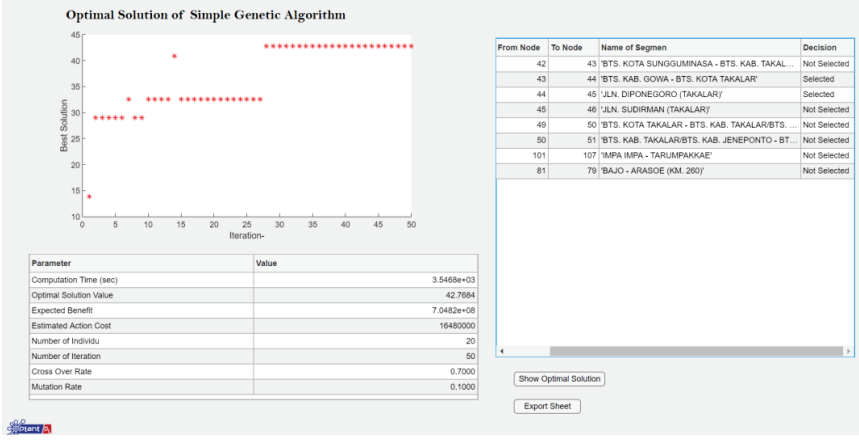


Gambar 13.8 Tangkapan layar solusi terbaik pada setiap iterasinya.

Hasil terkait penyelesaian permasalahan optimasi dapat diakses pada menu ***Result > Optimal Solution > PSO*** *dan* ***Result > Optimal Solution > GA***. Selain itu, hasil dalam bentuk numerik dapat diperoleh dengan menekan tombol ***Export Sheet*** yang akan mengekspor hasil dalam bentuk *spread sheet* pada lokasi penyimpanan *database* dan *map*. Adapun nama dokumen hasil tersebut adalah ***ResumeBestValueGA***, ***ResumeBestValuePSO***, ***ResumeDecisionPSO***, ***ResumeDecisionGA***.

13.3. Hasil Optimasi Studi Kasus

Setelah melakukan tahapan seperti yang disebutkan sebelumnya, OPTANT PL mampu memberikan solusi untuk permasalahan optimasi seperti yang terlihat pada Gambar 13.9 di bawah ini.



Gambar 13.9 Tangkapan layar solusi terbaik pada setiap iterasinya.

Dapat terlihat pada gambar tersebut, OPTANT PL memberikan rekomendasi untuk melebarkan ruas jalan BTS. Kab Gowa- BTS. Kota Talakar dan Jln. Diponegoro. Jalan ini pada kondisi tanpa adanya pelebaran jalan memiliki VCR 1,5 dan 1,1 yang menjadi salah satu alasan mengapa kedua segmen ini penting untuk diprioritaskan. Selain itu, kedua segmen ini memiliki biaya pelebaran yang cukup kecil, sehingga secara efisiensi pengunaan anggaran kedua segmen ini memiliki efisiensi yang tinggi. Meskipun dengan nilai VCR yang tinggi, memberikan tambahan alasan bahwa penggunaan anggaran secara efisien tetap mampu menghadirkan *benefit* yang paling besar pada jaringan tersebut. Oleh karenanya, OPTANT PL dengan metode GA merekomendasikan kedua segmen jalan tersebut.

Selain itu, gambar di atas memberikan ilustrasi pula bagaimana GA memperoleh solusi tersebut sepanjang iterasi. GA terlihat tidak stabil dalam proses pencarian solusi, dimana pada iterasi 15, GA berhasil memperoleh solusi yang cukup baik. Akan tetapi pada iterasi selanjutnya solusi ini hilang, sesuatu yang amat mungkin terjadi dalam konteks seleksi dalam prosedur GA. Atas dasar ini pula, banyak penelitian telah dikembangkan terkait prosedur elit pada GA untuk menghindari hilangnya solusi yang baik dalam proses seleksi. Secara konseptual prosedur elit berusaha menyimpan beberapa solusi baik untuk diloloskan menuju ke generasi selanjutnya tanpa melewati proses seleksi.

Referensi

Abdulaal, M., and LeBlanc, L.J. (1979) Continuous equilibrium network design models, Transportation Research Part B Vol.13, No. 1, pp. 19–32.

Bazaraa, M. S., Jarvis, J.J., Sherali, H.D., (1977). *Linear programming* and Network Flows. New York: John Wiley and Sons.

Boyce, D.E., and Janson, B.N. (1980) A discrete transportation network design problem with combined trip distribution and assignment, Transportation Research Part B, Vol. 14B, pp. 147–154.

Breedam, A.V. (2001) Comparing descent heuristics and metaheuristics for the vehicle routing problem, Computers & Operations Research Vol. 28, No. 4, pp. 289–315.

Cantarella, G.E., and Vitetta, A. (2006) The multi-criteria road network design problem in an urban area, Transportation Vol. 33, pp. 567–588.

Chen, M., and Alfa, A.S. (1991) A network design algorithm using a stochastic incremental traffic assignment approach, Transportation Science Vol. 25, No. 3, pp. 215– 224.

Chiou, S.W. (2005) Bilevel programming for the continuous transport network design problem, Transportation Research Part B Vol. 39, No. 4, pp. 361–383.

Chiou, S. (2008) A hybrid approach for optimal design of signalized road network, Applied Mathematical Modelling Vol. 32, No. 2, pp. 195–207.

Clausen, J. (1999) Branch and Bound Algorithms-Principles and Examples. citeseer.ist.psu.edu/683497.html

Coppin, B. (2004) Artificial Intelligence Illuminated, Jones Bartlett Illuminated Series. Sudbury, MA: Jones Bartlett Publishers.

Dantzig, G. B. and Ramser, J.H. (1959). The Truck Dispatching Problem. Management Science 6 (1), pp. 80–91.

Drezner, Z., and Wesolowsky, G.O. (1997) Selecting an optimum configuration of one-way and two-way routes, Transportation Science Vol. 31, No. 4, pp. 386–394.

Drezner, Z., and Salhi, S. (2000) Selecting a good configuration of one-way and two-way routes using tabu search, Control and Cybernetics Vol. 29, No. 3, pp. 725–740.

Drezner, Z., and Salhi, S. (2002) Using hybrid metaheuristics for the one-way and two-way network design problem, Naval Research Logistics (NRL) Vol. 49, No. 5, pp. 449–463.

Drezner, Z., and Wesolowsky, G.O. (2003) Network design: Selection and design of *link*s and facility location, Transportation Research Part A, Vol. 37, No. 3, pp. 241–256.

Eberhart, R. C., and Kennedy, J., 1995. A new optimiser using particle *swarm* theory. Proceedings of the Sixth International Symposium on Micro Machine and Human Science 43, 39–43.

Farahani, R.Z., Miandoabchi, E., Szeto, W.Y., Rashidi, H. (2013) A review of urban transportation network design problems, European Journal of Operational Research 229, pp. 281–302.

Garey, M.R. and Johnson, D.S. (1979) Computers and Intractability: A Guide to the Theory of NP-Completeness. Freeman.

Glover, F., and Kochenberger, G.A. (eds.) (2003) Handbook of Metaheuristics, Boston: Kluwer Academic Publishers, xi–xii.

Griffis, S., Bell, J.E., Closs, D.J. (2012) Metaheuristics in logistics and supply chain management, Journal of Business Logistics Vol. 33, No.2, pp. 90–106

Holland, J. H., 1975, Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor, MI.

Kennedy, J., and Eberhart, R. C. (1997) A discrete binary version of the particle *swarm* algorithm, Proceedings of the IEEE International Conference on Systems, Man and Cybernetics 5, pp. 4104–4108.

Krishnanand, K.N., Ghose, D., 2005. Detection of Multiple Source Locations using a *Glowworm* Metaphor with Applications to Collective Robotics. *Swarm* Intelligence symposium. In: Proceedings 2005 IEEE *Swarm* Intelligence Symposium, 2005, pp. 84–91

Krishnanand, K.N., Ghose, D., 2008. Theoretical Foundations for Rendezvous of *Glowworm*-Inspired Agent *Swarms* at Multiple Locations. Robotics and Autonomous Systems, Volume 56(7), pp. 549–569

Kuhn, H. W. (1955) 'The Hungarian Method for the Assignment Problem', *Naval Research Logistics Quarterly banner*, 2(1–2), pp. 83–97. doi: 10.1017/CBO9781107415324.004.

Land, H. and Doig, A. G. (1960) An automatic method of solving discrete programming proble*Ms.* Econometrica 28 (3), pp. 497–520.

Laporte, G. (1992). The Vehicle Routing Problem: An overview of exact and approximate algorith*Ms.* European Journal of Operational Research 59, pp. 345-358.

LeBlanc, L.J. (1975) An algorithm for the discrete network design problem, Transportation Science Vol. 9, No. 3, pp. 183–199.

Lee, C., and Yang, K. (1994) Network design of one-way streets with simulated annealing, Papers in Regional Science Vol. 73, No. 2, pp.119–134.

Lewis, C., (2008). *Linear programming*: Theory and applications. Whitman College Mathematics Department, 2008 - whitman.edu

Luathep, P., Sumalee, A., William, H.K.L., Li, Z.C., Lo, H.K. (2011) Global optimization method for mixed transportation network design problem: A mixed-integer *linear programming* approach, Transportation Research Part B Vol. 45, No. 5, pp. 808–827.

Long, J., Gao, Z., Zhang ,H., Szeto, W.Y. (2010) A turning restriction design problem in urban road networks, European Journal of Operational Research Vol. 206, No. 3, pp. 569–578.

Romanycia, M.H.J. and Pelletier, F.J. (1985), "What is a heuristic?", *Computational Intelligence*, Vol. 1 No. 1, pp. 47–58

Salman, A., Ahmad, I., Al-Madani, S., 2002. *Particle Swarm Optimisation* for task assignment problem. Microprocessors and Microsystems 26, 363–371.

Shen, Q., Jiang, J.H., Jiao, C.X., Shen, G.L, Yu, R.Q. (2004) Modified particle *swarm* optimization algorithm for variabel selection in MLR and PLS modeling: QSAR studies of antagonism of angiotensin II antagonists, European Journal of Pharmaceutical Sciences Vol. 22, No. 2–3, pp. 145–152.

Shapiro, J.F. (2001) Modeling the Supply Chain, Pacific Grove, CA: Thomson Learning Inc.

Shen, Q., Mei, Z., Ye, BX. (2009) Simultaneous genes and training samples selection by modified particle *swarm* optimization for gene expression data classification, Computers in Biology and Medicine, Vol. 39, No. 7, pp. 646 – 649.

Shi, Y., and Eberhart, R. (1998) A modified particle *swarm* optimiser, Proceedings of the

IEEE Conference on Evolutionary Computation, pp. 69–73.

Silver, E.A. (2004). An Overview of Heuristic Solution Methods. The Journal of the Operational Research Society, Vol. 55, No. 9, pp. 936-956.

Snyman, J.A., 2006. Practical Mathematical Optimization: An Introduction to Basic Optimization Theory and Classical and New Gradient-Based Algorith*Ms.* Springer Verlag

Steenbrink, A. (1974) Transport network optimization in the Dutch integral transportation study, Transportation Research Part B, Vol. 8, pp. 11–27.

Taha, H.A., "Operations Research, an Introduction", Prentice-Hall International, Inc., 8th Ed. (2007)

Talbi, E.G. (2009) Metaheuristics: From Design to Implementation. John Wiley & Sons

Menhas, M.I., Wang, L., Fei, M., Pan, H., (2012) Comparative performance analysis of various binary coded PSO algorithms in multivariabel PID controller design, Expert Systems with Applications Vol. 39, No. 4, pp. 4390–4401.

Menhas, M. I., Wang, L., Fei, M., Ma, C. (2011) Coordinated controller tuning of a boiler turbine unit with new binary *Particle Swarm Optimisation* algorithm, International Journal of Automation and Computing, Vol. 8, No. 2, pp. 185–192.

Menhas, M.I., Fei, M., Wang, L., Qian, L. (2012b) Real/binary co-operative and co-evolving *swarms* based multivariabel PID controller design of ball mill pulverizing system, Energy Conversion and Management Vol. 54, pp. 67–80.

Miandoabchi, E., Farahani, R.Z., Szeto, W.Y. (2012b) Bi-objective bimodal urban road network design using hybrid metaheuristics, Central European Journal of Operations Research Vol. 20, No 4, pp. 583–621.

Mitchell, J.E. (2000). Branch-and-Cut Algorithms for Combinatorial Optimization Proble*Ms.* The Handbook of Applied Optimization, Oxford University Press.

Munkres, J. (1957) 'Algorithms for the assignment and transportation problems', *Journal of the Society for Industrial and Applied Mathematics*, 5(1), pp. 32–38.

Poorzahedy H. (1980) Efficient algorithms for solving the network design problem, Ph.D. Dissertation, Northwestern University, Department of Civil Engineering, Evanston, Illinois.

Poorzahedy, H., and Turnquist, M.A. (1982) Approximate algorithms for the discrete network design problem, Transportation Research Part B Vol. 16, No. 1, pp. 45–55.

Poorzahedy, H., and Abulghasemi, F. (2005) Application of ant system to network design problem, Transportation Vol. 32, No. 3, pp. 251–273.

Poorzahedy, H., and Rouhani, O. (2007) Hybrid meta-heuristic algorithms for solving network design problem, European Journal of Operation Research Vol. 182, pp. 578–596.

Wu, J.J., Sun, H.J., Gao, Z.Y., Zhang, H.Z. (2009) Reversible lane-based traffic network optimization with an advanced traveller information system, Engineering Optimization 41, pp. 87–97.

Wang, L., Wang, X., Fu, J., Zhen, L. (2008) A novel probability binary particle *swarm* optimization algorithm and its application, Journal of Software Vol. 9, No. 3, pp. 28–35.

Xiong, Y., and Schneider, J.B. (1992) Transportation network design using a cumulative algorithm and neural network, Transportation Research Record 1364, pp. 37–44.

Yamada, T. Russ, B.F., Castro, J. Taniguchi, E. (2009) Designing multimodal freight transport networks: A heuristic approach and applications, Transportation Science Vol. 43, No. 2, pp. 129–143.

Yamada, T., Febri, Z., 2015. Freight transport network design using *Particle Swarm Optimisation* in supply chain–transport supernetwork equilibrium. *Transportation Research Part E: Logistics and Transportation Review* 75, pp. 164–187.

Yang, H. (1997) Sensitivity analysis for the elastic-demand network equilibrium problem with applications. Transportation Research Part B Vol. 31, No. 1, pp. 55–70.

Yang, H., and Bell, M.G.H. (1998) Models and algorithms for road network design: A review and some new developments, Transport Reviews Vol. 18, No. 3, pp. 257–278.

Yang, H., and Meng, Q. (2000) Highway pricing and capacity choice in a road network under a build–operate–transfer scheme, Transportation Research Part A Vol. 34, pp. 207–222.

Ziyou, G.,Yifan, S. (2002) A reserve capacity model of optimal signal control with user-equilibrium route choice, Transportation Research Part B Vol. 36, No. 4, pp. 313–323.

Zhang, H., and Gao, Z. (2007) Two-way road network design problem with variabel lanes, Journal of Systems Science and Systems Engineering, Vol. 16, No. 1, pp. 50–61.

Zukhruf, F., Frazila, R. B. and Widhiarso, W. (2020) 'A comparative study on *swarm*-based algorithms to solve the stochastic optimization problem in container terminal design', *International Journal of Technology*, 11(2), pp. 374–387. doi: 10.14716/ijtech.v11i2.2090.

Zukhruf, F., Yamada, T., Taniguchi, E. (2014) Designing Cocoa Transport Networks Using a Supply Chain Network Equilibrium Model with the Behaviour of Freight Carriers. Journal of Japan Society of Civil Engineers, Ser. D3 (Infrastructure Planning and Management.

PENGANTAR OPTIMASI
DALAM REKAYASA TRANSPORTASI

Buku ini pada dasarnya hanya sebuah kumpulan kecil dari luasnya ilmu pengetahuan di bidang optimasi yang berserak di jagad semesta. Bagian-bagian yang berserak tersebut kemudian berusaha dikontekstualisasi ke dalam permasalahan-permasalahan sederhana pada disiplin keilmuan rekayasa transportasi. Proses ini menjadi penting seiring semakin tidak dapat dipisahkannya penerapan prinsip optimasi dalam pengambilan keputusan di bidang rekayasa transportasi. Sehingga buku ini diharapkan menjadi salah satu pengantar untuk melakukan eksplorasi secara luas terhadap prinsip dan teknik optimasi dalam pengambilan keputusan di bidang rekayasa transportasi.

Sebagai pengantar dalam memahami optimasi, buku ini membahas fitur utama dari optimasi yang dibahas secara iteratif pada banyak kasus terkait rekayasa transportasi. Metode penyelesaian permasalahan optimasi pun dijelaskan secara runtut dan aplikatif, disertai dengan contoh dan penggunaan alat bantu perhitungan. Proses ini diharapkan dapat menjadi sarana untuk mengantarkan transformasi pembaca dari kemampuan memahami sebuah konsep menjadi sebuah kemampuan dalam penyelesaian masalah. Video penjelasan pada materi-materi utama pun diberikan dalam buku ini, sebagai bagian dari usaha memberikan pemahaman lebih baik kepada pembaca. Buku ini sangat cocok dibaca oleh mahasiswa maupun praktisi yang ingin memahami secara konseptual maupun teknikal penerapan optimasi dalam rekayasa transportasi.

ITB Press
Gedung Perpustakaan Pusat ITB
Lantai Basement, Jl. Ganesa No. 10
Bandung 40132, Jawa Barat
Telp. 022 2504257/022 2534155
e-mail: office@itbpress.itb.ac.id
web: www.itbpress.itb.ac.id
Anggota Ikapi No. 034/JBA/92
APPTI No. 005.062.1.10.2018

Seratus Satu Tahun
Institut Teknologi Bandung

ISBN 978-623-297-155-4