UNIVERSITAS INDONESIA

OPTIMASI JARINGAN PIPA UNTUK DISTRIBUSI GAS MENGGUNAKAN ALGORITMA GENETIK

TESIS

YURITA PUJI AGUSTIANI
09 06 57 90 33

FAKULTAS TEKNIK
PROGRAM STUDI TEKNIK KIMIA
PROGRAM MAGISTER MANAJEMEN GAS
JAKARTA
JANUARI 2011

UNIVERSITAS INDONESIA

OPTIMASI JARINGAN PIPA UNTUK DISTRIBUSI GAS MENGGUNAKAN ALGORITMA GENETIK

TESIS

Diajukan sebagai salah satu syarat untuk memperoleh gelar Magister Teknik

YURITA PUJI AGUSTIANI
09 06 57 90 33

FAKULTAS TEKNIK
PROGRAM STUDI TEKNIK KIMIA
PROGRAM MAGISTER MANAJEMEN GAS
JAKARTA

JANUARI 2011

HALAMAN PERNYATAAN ORISINALITAS

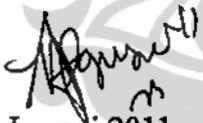
Tesis ini adalah hasil karya saya sendiri,

dan semua sumber baik yang dikutip maupun dirujuk

telah saya nyatakan dengan benar

Nama : Yurita Puji Agustiani

NPM : 0906579053

Tanda Tangan : 

Tanggal : Januari 2011

HALAMAN PENGESAHAN

Tesis ini diajukan oleh :

Nama	:	Yurita Puji Agustiani
NPM	:	0906579033
Program Studi	:	Teknik Kimia bidang kekhususan Manajemen Gas
Judul Tesis	:	Optimasi Distribusi Gas pada Pipa Menggunakan Algoritma Genetik

Telah berhasil dipertahankan di hadapan Dewan Penguji dan diterima sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Magister Teknik pada Program Studi Teknik Kimia bidang kekhususan Manajemen Gas, Fakultas Teknik, Universitas Indonesia.

DEWAN PENGUJI

Pembimbing 1 : **Dr. Ir. Mahmud Sudibandriyo, MSc** (.....)

Penguji 1 : **Dr.rer.nat.Ir.Yuswan Muharam, MT** (.....)

Penguji 2 : **Dr.Ir.Asep Handaya S, M.Eng** (.....)

Penguji 3 : **Kamarza Mulia, PhD** (.....)

Ditetapkan di : Jakarta

Tanggal : Januari 2011

KATA PENGANTAR

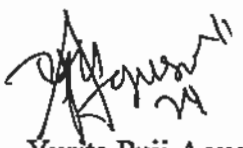
Puji syukur kepada Allah SWT, karena atas berkat rahmat-Nya, tesis ini dapat diselesaikan. Penulisan tesis ini dilakukan dalam rangka memenuhi salah satu syarat untuk mencapai gelar Magister Teknik Program Studi Teknik Kimia pada Fakultas Teknik Universitas Indonesia. Saya menyadari bahwa dari masa perkuliahan hingga penyususan tesis ini, telah banyak pihak yang membantu sehingga semua proses dapat berjalan dengan baik. Oleh karena itu, saya mengucapkan terimakasih dengan tulus kepada:

1. Bapak Bapak Ir. Mahmud Sudibandrio, MSc., PhD selaku dosen pembimbing yang telah menyediakan waktu, tenaga dan pikiran untuk mengarahkan saya dalam penyusunan tesis ini.
2. Seluruh staf pengajar Pasca Sarjana Magister Manajemen Gas Universitas Indonesia
3. Seluruh pihak pihak yang telah bersedia menjadi nara sumber baik menjadi responden maupun para pakar dalam penelitian ini
4. Keluarga tersayang yang telah membantu dengan doa yang tulus.
5. Teman-teman S2 atas kerjasama dalam menyelesaikan tugas dan tesis
6. Pihak pihak lain yang tidak dapat disebut satu persatu.

Penulis menyadari akan keterbatasan kemampuan dan wawasan dalam penyusunan tesis ini sehingga segala kritik dan saran yang bermanfaat diharapkan dapat memperbaiki penelitian ini di masa mendatang.

Akhir kata, Saya berharap Allah SWT berkenan membalas segala kebaikan semua pihak yang telah membantu. Semoga tesis ini membawa manfaat.

Jakarta, Januari 2011



Yurita Puji Agustiani

HALAMAN PERNYATAAN PERSETUJUAN PUBLKASI TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

Sebagai civitas akademik Universitas Indonesia, saya yang bertanda tangan dibawah ini :

Nama	: Yurita Puji Agustiani
NPM	: 0906579033
Program Studi	: Manajemen Gas
Departemen	: Teknik Kimia
Fakultas	: Teknik
Jenis Karya	: Tesis

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

"OPTIMASI JARINGAN PIPA UNTUK DISTRIBUSI GAS MENGGUNAKAN ALGORITMA GENETIK"

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

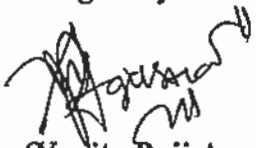
Saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Jakarta

Pada tanggal : Januari 2011

Yang menyatakan



(Yurita Puji Agustiani)

ABSTRAK

Nama : Yurita Puji Agustiani
Program Studi : Teknik Kimia bidang kekhususan Manajemen Gas
Judul : OPTIMASI JARINGAN PIPA UNTUK DISTRIBUSI GAS MENGGUNAKAN ALGORITMA GENETIK

Tesis ini membahas mengenai optimasi jaringan pipa untuk distribusi gas. Nilai investasi yang terkecil merupakan tujuan dari optimasi. Hal ini dapat diperoleh dengan menggunakan diameter pipa yang paling sesuai untuk jaringan pipa dengan total *cost* yang paling rendah. Algoritma genetik merupakan salah satu metode yang dapat digunakan untuk melakukan optimasi.

Kata Kunci:
Optimasi, Jaringan Pipa untuk Distribusi Gas, Algoritma Genetik

ABSTRACT

Name : Yurita Puji Agustiani
Study Program: Chemistry Technique, Speciality Area Gas Management
Title : OPTIMIZE PIPELINE NETWORK FOR GAS DISTRIBUTION USING GENETIC ALGORITHM

This thesis discusses the optimization of gas pipelines for distribution. The smallest investment value is the purpose of optimization. This can be obtained by using the most appropriate pipe diameter for the pipeline network and with the total lowest *cost*. Genetic algorithms are one of the methods that can be used to perform optimization.

Keywords:
Optimize, Pipeline Network for Gas Distribution, Genetic Algorithm (GA)

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB 1

PENDAHULUAN

1.1 Latar Belakang

Penggunaan bahan bakar minyak di Indonesia sangat besar untuk penggunaan dalam rumah tangga dan BBM. Subsidi pada BBM sedikit demi sedikit akan dihapuskan sesuai dengan keputusan bersama antara pemerintah dan DPR yang tertuang pada UU No. 25/2000 tentang Program Pembangunan Nasional. Sedangan subsidi untuk minyak tanah sebagai penunjang kehidupan sehari-hari yang harga jualnya rendah dibandingkan dengan biaya produksinya yang mahal masih di suplai/didukung oleh pemerintah. Krisis dalam managemen energi nasional dan harga minyak dunia yang mahal membuat pemerintah merasa terlalu berat dalam menyuplai besarnya subsidi yang harus ditanggung dalam memenuhi kebutuhan masyarakat terhadap minyak tanah. Nilai subsidi untuk minyak tanah sangat besar sehingga bisa memperlambat pembangunan. Jika nilai subsidi ini bisa ditekan atau dihapuskan tentu nilai RAPBN untuk subsidi minyak tanah bisa dialihkan untuk pembangunan.

Solusi yang diambil pemerintah untuk mengatasi masalah ini yaitu mengkonversi penggunaan bahan bakar minyak tanah ke bahan bakar gas. Penggunaan BBG akan memberikan dampak yang positif bagi masyarakat karena selain lebih ramah lingkungan, lebih murah, dan lebih aman juga sebagai bentuk pengurangan penggunaan minyak bumi. Sedangkan keuntungan bagi pemerintahan dapat mengurangi beban subsidi yang bisa dialihkan untuk biaya pembangunan.

Rencana pemerintah dalam mengkonversi penggunaan minyak tanah ke bahan bakar gas didukung dengan kesediaan gas bumi dalam negeri yang cukup besar serta perencanaan pembangunan gas kota. Penggunaan LPG juga merupakan solusi untuk konversi penggunaan BBM menjadi BBG, harga LPG lebih murah daripada minyak tanah namun LPG masih mahal jika dibandingkan dengan gas bumi. Dengan adanya gas kota maka akan mengurangi penggunaan LPG dan lebih aman mengingat banyaknya tabung-tabung LPG yang meledak sehingga meresahan

masyarakat. Solusi gas kota ini dapat mempermudah masyarakat dalam menggunakan BBG untuk kebutuhan sehari-hari sehingga berdampak pada meningkatnya jumlah masyarakat yang menggunakan gas. Pemerintah membuat kebijakan-kebijakan untuk mendukung peningkatan penggunaan gas melalui UU No 22 Tahun 2001 tentang minyak dan gas bumi yang didalamnya berisi keinginan untuk meningkatkan penggunaan gas, Peraturan pemerintah no 36 tahun 2004 mengenai kegiatan Usaha Hilir Migas yang merupakan implementasi dari UU No 22 tahun 2001, kebijakan Energi Nasional 2003-2020 tanggal 24 Februari 2004 tetang pembangunan infrastruktur gas bumi. Namun sayangnya dalam proses pembangunan infrastruktur jaringan pipa gas terbentur pada keterbatasan dana yang dimiliki pemerintah. (www.bpkp.go.id)

Pembangunan infrastuktur ditribusi gas bumi direncanakan dengan pipanisasi. Minimalisasi biaya pembangunan infrastuktur perlu dlakukan agar pemerintah mampu membiayai pembangunan gas kota. Bagian yang cukup besar dalam infrastuktur jaringan pipa gas adalah penggunaan pipa. Penggunaan pipa yang minimal tetapi menunjang penyebaran gas suatu kota tentu berbanding lurus dengan pengeluaran minimal yang diperlukan dalam membangun gas kota.

Suatu kota memiliki beberapa wilayah yang perlu disuplai gas. Dengan mengoptimasi penggunaan pipa dari pusat gas ke wilayah - wilayah tersebut diharapkan bisa meminimalkan biaya penggunaan pipa.

Dijaman sekarang ini, permintaan untuk menurunkan harga produksi selalu bertambah. Hal ini dikarenakan kompetisi market global yang makin sulit. Kebutuhan akan engineer untuk dapat menemukan cara menurunkan harga produksi ini makin meningkat. Salah metode yang digunakan untuk menurunkan harga produksi ini adalah melalui metode optimasi. Metode optimasi ini dapat digunakan untuk mengefisiensikan dan menurunkan harga produksi. Proses optimasi diameter pipa dapat dilakukan untuk meminimalkan biaya investasi dengan mempertimbangkan tingkat tekanan dan aliran yang telah disepakati dalam kontrak.

Metode optimasi telah mencapai titik dimana telah melalui uji coba dan diimplementasikan di berbagai industri seperti otomotif, farmasi, konstruksi, dan lain sebagainya. Seiring dengan meningkatnya teknologi komputer, permasalahan yang

dapat diselesaikan juga semakin meningkat. Komputer merupakan alat pendukung dari metode optimasi yang tidak dapat dipisahkan.

Studi tentang optimasi perpipaan telah dilakukan oleh Prof. Dr. Septoratno Siregar dkk dari ITB yang membahas mengenai desain pipa transmisi yang optimum untuk laju pengiriman gas pada jaringan pipa, sehingga biaya yang dikeluarkan minimum namun sesuai persyaratan dengan menggunakan metode heuristik pada software OPPINET. Program untuk kasus penentuan penggunaan diameter pipa di lakukan oleh Omar Fayez Mohamed dari Faculty of Engineering di Cairo University, Egypt. Program yang di bangun hanya terbatas untuk satu skema jaringan perpipaan dengan menggunakan Algoritma Genetik. Optimasi dengan menggunakan Algoritma Genetik juga dilakukan oleh Hosaam A. A. Abdel –Gawad dari Mansoura University, Egypt untuk mengotimasi desain perpipaan untuk masalah penentuan jalur pipa dengan membangun program FORTRAN. Dalam penelitian ini saya akan menerapkan Genetik Algoritma untuk mengoptimasi diameter pipa pada jaringan perpipaan. Optimasi ini diharapkan dapat meminimumkan biaya investasi untuk penggunaan pipa. Program yang akan dibangun memiliki flesibilitas untuk berbagai skema jaringan dengan ketentuan yang terdapat pada batasan masalah yang akan dijelaskan di sub-bab berikutnya.

1.2 Rumusan Penelitian

Rumusan masalah dari tesis ini adalah sebagai berikut:

- Bagaimana meminimalkan biaya penggunaan pipa dengan mengoptimasi diameter pipa yang digunakan dalam suatu jaringan distribusi gas dengan spesifikasi *pressure* dan *Demand Flow rate* yang telah ditentukan
- Berapa perkiraan biaya dalam membangun jaringan distribusi gas berdasarkan pada hasil optimasi diameter pipa.

1.3 Tujuan Masalah

Membangun perangkat lunak untuk perhitungan optimasi secara kasar penggunaan pipa sebagai kebutuhan pembangunan infratruktur distribusi gas kota dengan menggunakan Algoritma Genetik. Diharapkan hasil dari perhitungan

perangkat lunak ini dapat memberikan keputusan pilihan yang mendekati benar dalam mengestimasi biaya pembangunan infratruktur di suatu kota.

1.4 Batasan Penelitian

Batasan masalah dari tesis ini adalah sebagai berikut:

- Studi kasus yang dipilih adalah optimasi pada perpipaan
- Hanya satu teknik optimasi yg akan dipilih utk menyelesaikan kasus ini yaitu Algoritma Genetik
- Algoritma Genetik yang dipilih adalah Algoritma Genetik biner
- Simulasi yang akan dilakukan memiliki batasan-batasan diantaranya
  a. Pada jaringan tidak menggunakan alat-alat bantuan seperti kompresor dan lain-lain
  b. Pipa yang digunakan dalam jaringan yaitu *carbon stell*
  c. Yang dialirkan dalam jaringan ini adalah gas alam (1 fasa)
  d. Jaringan *path* diasumsikan lurus
  e. Untuk setiap *path* menggunakan satu ukuran
  f. Tidak ada *loop* dalam skema jaringan
  g. Setiap jaringan hanya terdapat satu *source*
  h. Suhu dianggap konstan
  i. Faktor ketinggian diabaikan

1.5 Sistematika Penulisan

Dalam penulisan tesis ini dibagi dalam beberapa bab dan sub bab dengan perincian lengkap seperti pada daftar isi. Secara ringkas dapat disebutkan sebagai berikut :

BAB 1 merupakan bab pendahuluan yang berisikan latar belakang, perumusan masalah, tujuan penulisan, batasan masalah, dan sistematika penulisan.

BAB 2 merupakan bab landasan teori yang membahas tentang teori yang berkaitan dengan perhitungan perpipaan gas dan parameter-parameter yang terkait dalam perpipaan.

BAB 3 merupakan bab metode penelitian yang berisi pembahasan mengenai penerapan Algoritma Genetik dengan kasus optimasi diamter pipa dan kerangka program yang akan dibangun.

BAB 4 merupakan bab simulai program dan hasil optimasi dari suatu contoh kasus.

BAB 5 merupakan bab kesimpulan dan saran dari hasil penulisan secara keseluruhan. Dalam lembaran akhir dicantumkan lampiran-lampiran lain yang menunjang isi bab-bab sebelumnya.

BAB 2

DASAR TEORI

2.1 Optimasi

Optimasi merupakan proses untuk memperoleh hasil yang terbaik dari persyaratan yang ada. Berikut definisi optimasi dari beberapa literatur yang diperoleh:

Berdasarkan Rao [8],"Optimisasi dapat didefinisikan sebagai proses untuk mencari kondisi yang memberikan nilai maksimum atau minimum dari sebuah fungsi". Fungsi disini merupakan hasil pemodelan secara matematis dari situasi keputusan yang dihadapi.

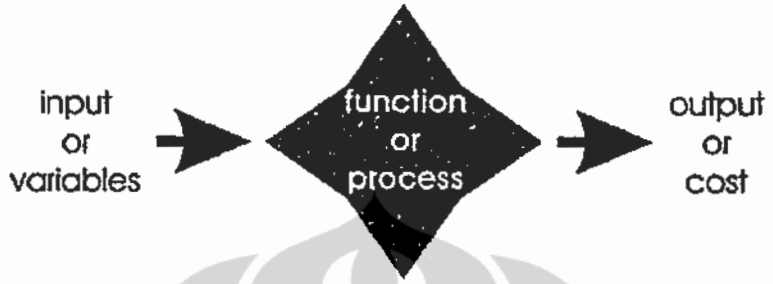
Definisi lain oleh Fletcher [3],"Optimasi dapat didefinisikan sebagai ilmu untuk menentukan solusi 'terbaik' untuk sebuah permasalahan matematika, yang umumnya dimodelkan sebagai kenyataan fisik". Dari definisi ini kita dapat menyimpulkan bahwa tujuan dari optimasi ini adalah untuk meminimalisasikan 'usaha' yg diperlukan dan memperoleh keuntungan semaksimal mungkin.

Definisi lain oleh Haupt [5], "Optimasi adalah proses untuk membuat sesuatu menjadi lebih baik. Optimasi melibatkan pencobaan terhadap variasi-variasi pada konsep yang ditentukan awal dan menggunakan informasi yang diperoleh untuk mempertajam data. Komputer merupakan peralatan yang sempurna untuk permasalahan optimasi selama ide atau variabel yang mempengaruhinya dapat di *input* dalam format elektronis."

Dalam optimasi, tidak ada satu teknik optimasi yang dapat memecahkan semua permasalahan yang ada. Namun beberapa teknik optimasi dikembangkan untuk memecahkan masalah-masalah tertentu. Optimasi merupakan bagian dari *operations research*. *Operation research* sendiri merupakan cabang dari matematika yang memperhatikan pengaplikasian dari metode ilmiah dan teknik untuk memecahkan permasalahan pengambil keputusan. [8]

Teknik yang ada dalam optimasi ini terlalu luas, sehingga hanya satu teknik yang akan dipilih untuk kasus ini yaitu algoritma genetik.

Gambar 2.1 menggambarkan proses dari optimasi secara garis besar, optimasi akan menyesuaikan *input* sehingga diperoleh hasil yang minimum atau maksimum. Pada gambar tersebut, *input* terdiri dari variabel-variabel, fungsi atau proses dikenal juga sebagai fungsi *cost*, output merupakan *cost* atau *fitness*.



Gambar 2.1 Proses Optimasi secara Umum [8]

2.2 Algoritma Genetik

2.2.1 Pendahuluan Algoritma Genetik

Algoritma genetik merupakan teknik optimasi dan pencarian yang berdasarkan terhadap prinsip-prinsip genetik dan *natural selection.* Menurut Haupt [5], keuntungan dari penggunaan algoritma genetik yaitu:

- Optimasi dengan variabel kontinu atau diskrit
- Tidak memerlukan penurunan informasi
- Dapat secara paralel melakukan pencarian dari hasil *sampling cost surface*
- Jumlah variabel yang dapat ditangani cukup banyak
- Dapat memanfaatkan komputer yang menerapkan sistem paralel (beberapa komputer dijadikan seperti satu komputer)
- Optimasi variabel dengan *cost surface* yang kompleks (dapat lompat keluar dari local minimum)
- Menyediakan daftar dari variabel-variabel yang optimum (tidak hanya menghasilkan satu solusi)
- Dapat menggunakan variabel yang di-*encode* (enkripsi) dan optimasi dilakukan dengan variabel yang di-*encode* tersebut

Algoritma Genetik merupakan teknik optimasi dan pencarian yang berdasarkan prinsip-prinsip genetika dan seleksi alami (*natural selection*). Sebuah algoritma genetik memungkinkan sebuah populasi yang terdiri dari banyak individu,

untuk berevolusi dibawah aturan-aturan seleksi yang telah ditetapkan kedalam tahap yang dapat memaksimalkan "*fitness*" (meminimalkan fungsi *cost*). Metode ini dikembangkan oleh John Holland (1975), dan dipopulerkan oleh muridnya yaitu David Goldberg, dimana digunakan untuk menyelesaikan permasalahan sulit yang melibatkan kontrol dari transmisi *gas-pipeline*. [5]

2.2.2 Analogi Algoritma Genetik dengan Genetik pada Biologi

Algoritma genetik berkembang dari teori genetik pada biologi. Sehingga ada baiknya untuk mengenal beberapa istilah-istilahnya dan analoginya.

Tiap organisme terdiri dari sel, tiap sel mengandung sepasang ato lebih kromosom yang sama. Kromosom merupakan rangkaian DNA, yang berfungsi sebagai 'blueprint' *organism*. Secara konsep kromosom dapat dipecah menjadi gen, dengan masing-masing gen mengkodekan suatu jenis protein tertentu. [7]

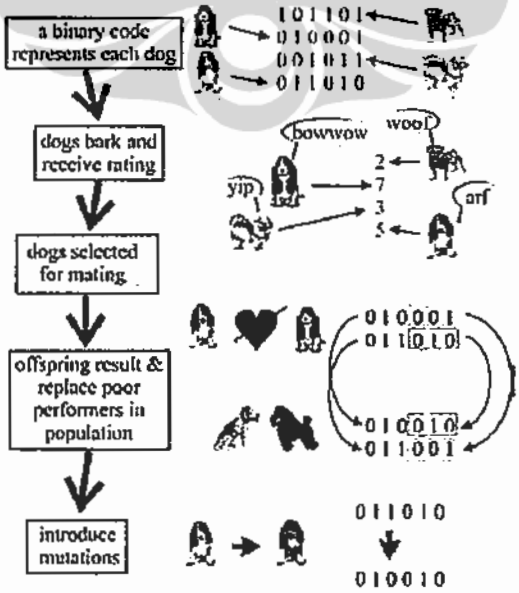
Organisme umumnya memiliki beberapa kromosom pada selnya. Kumpulan yang lengkap dari materi genetik (semua kromosom diambil jadi satu) dikenal sebagai *genome* dari organisme. Istilah *genotype* menunjukkan suatu set tertentu dari gen yang terkandung dalam *genome*. [7]

Organisme dimana kromosomnya tersusun dalam pasangan dikenal sebagai *diploid*. Sebaliknya jika kromosom tidak berpasangan dikenal sebagai *haploid*. Saat terjadi sebuah reproduksi seksual pada *diploid*, *recombination* (*crossover*) terjadi. Pada orang tuanya, gen bertukaran diantara tiap pasang kromosom untuk membentuk *gamete* (kromosom tunggal), kemudian *gamete* dari masing-masing orang tua berpasangan untuk membentuk suatu set lengkap dari kromosom *diploid*. Dalam reproduksi seksual *haploid*, gen kromosom dipertukarkan antara kedua orang tua tunggal-untai. Keturunannya akan dikenakan mutasi, di mana *nukleotida* tunggal (bit dasar DNA) akan mengalami perubahan dari induk ke keturunannya, sering kali perubahan yang dilakukan dihasilkan dari kesalahan penyalinan. Kesesuaian organisme biasanya didefinisikan sebagai probabilitas bahwa organisme akan hidup untuk mereproduksi (viabilitas) atau sebagai fungsi dari jumlah keturunan organisme yang dimiliki (kesuburan). [7]

Dalam algoritma genetika, istilah kromosom biasanya mengacu pada calon dari solusi permasalahan, dan sering dikodekan sebagai bit string. "gen" umumnya

berbentuk bit tunggal atau blok pendek dari bit yang berdekatan dan mengkodekan suatu elemen tertentu dari calon solusi. *Crossover* biasanya terdiri dari pertukaran bahan genetik antara dua orang tua tunggal kromosom *haploid*. Mutasi terjadi dengan membalikkan bit pada lokus yang dipilih secara acak. [7]

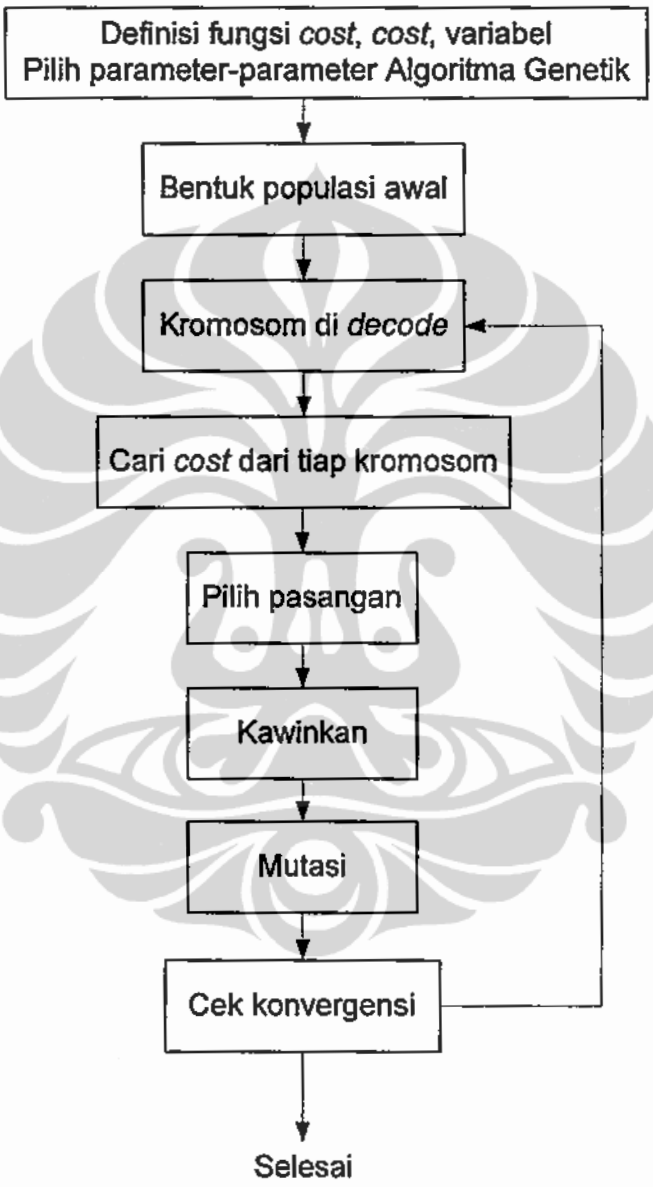
Gambar 2.2 menggambarkan analogi antara evolusi biologi dengan algoritma genetika tipe *binary*. Keduanya diawali dengan populasi awal dengan anggota secara acak. Tiap baris dari angka biner merepresentasikan satu anjing dari populasi tersebut. Sifat yang berhubungan dengan gonggongan yang keras di kodekan dengan urutan biner dan diasosiasikan dengan anjing tersebut. Jika kita akan membiakkan anjing dengan gonggongan terkeras, maka hanya beberapa anjing yang akan disimpan untuk dibiakkan. Harus ada suatu cara dalam menentukan kerasnya gonggongan, anjing dengan gonggongan lemah akan memperoleh nilai *cost* yang rendah. Kemudian dari populasi pembiakkan tersebut, 2 anjing dipilih secara acak untuk menurunkan 2 anak anjing baru. 2 anak anjing tersebut memiliki probalitas yang tinggi untuk menjadi anjing dengan gonggongan yang terkeras karena kedua orang tuanya memiliki gen yang dapat membuat mereka memiliki gonggongan terkeras. Urutan biner yang dari anak anjing tersebut mengandung bagian dari urutan biner orang tuanya. Anak anjing ini akan menggantikan anjing yang gonggongannya lemah pada populasi ini, sehingga jumlah populasi dalam keadaan yang awal. Iterasi dari kondisi ini akan menghasilkan anjing dengan gonggongan sangat keras.



Gambar 2.2 Analogi antara algoritma genetika numerik dan genetika biologi. [5]

2.2.3 Proses Kerja Algoritma Genetik

Gambar 2.3 merupakan *flowchart* dari sebuah algoritma genetik, tahapan-tahapan dalam *flowchart* tersebut akan dijelaskan lebih rinci. [5]



Gambar 2.3 *Flowchart* sebuah Algoritma Genetik

Alur kerja algoritma genetik berdasarkan [7] yaitu:

1. Dimulai dari populasi dengan kromosom *l*-bit yang dihasilkan secara acak (merupakan calon solusi dari permasalahan)

2. Hitung fungsi *fitness f(x)* dari tiap kromosom *x* pada populasi
3. Ulangi langkah-langkah berikut sampai *offspring n* telah terbentuk:
    a. Pilih sepasang kromosom orang tua dari populasi yang sekarang, probabilitas dari *selection* menjadi fungsi peningkatan dari fungsi *fitness*. *Selection* dilakukan dengan *replacement*, maksudnya dimana kromosom yang sama dapat dipilih lebih dari sekali untuk menjadi orang tua.
    b. Dengan probabilitas Pc (Probabilitas *Crossover* atau Laju *Crossover*), *crossover* pasangan dengan titik-titik yang dipilih secara acak (menggunakan probabilitas yang *uniform*) untuk membentuk dua *offspring*. Jika tidak ada *offspring* baru yang diperoleh, bentuk offspring baru berdasarkan salinan yang sama dari kedua orangtuanya
    c. Mutasi kedua *offspring* pada tiap locus dengan probabilitas P*m* (Probabilitas Mutasi atau Laju Mutasi), dan letakkan hasil dari kromosom pada populasi yang baru. Jika *n* adalah ganjil, satu anggota populasi dapat dibuang secara acak.
4. Ganti populasi yang sekarang dengan populasi yang baru
5. Lakukan langkah 2

2.2.3.1 Pemilihan Variabel dan Fungsi *Cost*

Sebuah fungsi *cost* membentuk keluaran dari suatu set variabel masukan (kromosom). Fungsi *cost* ini dapat berupa fungsi matematika, sebuah penelitian, atau sebuah permainan. Tujuannya adalah untuk memodifikasi keluaran dengan beberapa mode yang diinginkan dengan cara mencari variabel masukan dimana nilai-nilainya sesuai. Istilah *fitness* akan secara intensif digunakan untuk menunjukkan keluaran dari fungsi tujuan. [5]

Algoritma Genetik dimulai dengan mendefinisikan sebuah kromosom atau sebuah susunan nilai variabel yang untuk dioptimasi. Jika kromosom tersebut memiliki n variabel (permasalahan optimasi dengan n dimensi) berbentuk $p_1, p_2, ..., p_n$ maka kromosom tersebut dapat ditulis sebagai n elemen baris vektor, seperti yang ditunjukkan pada Persamaan 2.1. [5]

$$kromosom = [p_1, p_2, \dots, p_n] \quad \dots (2.1)$$

Tiap kromosom memiliki *cost* yang diperoleh dengan mengevaluasinya terhadap fungsi *cost f*, sehingga ditunjukkan pada Persamaan 2.2. [5]

$$cost = f(kromosom) = f(p_1, p_2, \dots, p_n) \quad \dots (2.2)$$

Sebagai contoh untuk jika kita mencari kedalaman laut yang terdalam maka fungsi *cost*-nya yaitu pada Persamaan 2.3. [5]

$$f(x, y) = kedalaman\ pada\ (x, y) \quad \dots (2.3)$$

2.2.3.2 *Encoding* dan *Decoding* Variabel

Karena nilai variabel direpresentasikan dalam biner, maka dibutuhkan suatu cara untuk mengkonversi nilai-nilai kontinu menjadi biner. Untuk *encoding n* variabel, P_n, ditunjukkan pada Persamaan 2.4 dan Persamaan 2.5. [5]

$$P_{norm} = \frac{P_n - P_{lo}}{P_{hl} - P_{lo}} \quad \dots (2.4)$$

$$gene[m] = round\left\{P_{norm} - 2^{-m} - \sum_{p=1}^{m-1} gene[p]2^{-p}\right\} \quad \dots (2.5)$$

Untuk *decoding*: [5]

$$P_{quant} = \sum_{m=1}^{N_{gene}} gene[m]2^{-m} + 2^{-(M+1)} \quad \dots (2.6)$$

$$Q_n = P_{quant}(P_{hl} - P_{lo}) + P_{lo} \quad \dots (2.7)$$

Dimana:

P_{norm}	= Variabel yang dinormalisasi, $0 \leq P_{norm} \leq 1$
P_{lo}	= Nilai variabel yang terkecil
P_{hl}	= Nilai variabel yang terbesar
$gene[m]$	= Versi biner dari P_n
$round\{\cdot\}$	= Pembulatan ke integer terdekat
P_{quant}	= Versi kuantisasi dari P_{norm}
Q_n	= Versi kuantisasi dari P_n

Nilai kuantisasi dari gen atau variabel dapat ditemukan dengan cara mengalikan vektor yang mengandung bit-bit dengan vektor yang mengandung tingkatan kuantisasi, ditunjukkan pada Persamaan 2.8. [5]

$$Q_n = gene \times Q^T \quad \ldots (2.8)$$

Dimana:

$gene$ $= \left[b_1 b_2 \ldots b_{N_{gene}}\right]$

N_{gene} = Jumlah bit pada sebuah gen

b_n = Bit biner = 1 atau 0

Q = Vektor kuantisasi = $[2^{-1}2^{-2} \ldots 2^{N_{gene}}]$

Q^T = Transpose dari Q

2.2.3.3 Populasi

Algoritma genetik dimulai dengan sejumlah kromosom yang dikenal sebagai populasi. Populasi tersebut memiliki kromosom N_{pop} dan merupakan matriks $N_{pop} \times N_{bits}$yang dipenuhi dengan angka acak 0 dan 1, dihasilkan dari Persamaan 2.9. Haupt (2004).

$$pop = round\left(rand(N_{pop}, N_{bits})\right) \quad \ldots (2.9)$$

2.2.3.4 Seleksi Alam

Kelangsungan hidup dapat dianalogikan membuang kromosom dengan nilai *cost* tertinggi. *Cost* dari N_{pop} dan asosiasi kromosomnya akan diurutkan dari *cost* terendah sampai tertinggi. Kemudian hanya yang terbaik yang akan berlanjut dan sisanya dihapus. Persamaan 2.10 menunjukkan jumlah kromosom yang akan disimpan dalam tiap generasinya. [5]

$$N_{keep} = X_{rate} N_{pop} \quad \ldots (2.10)$$

Seleksi alam ini akan terjadi dalam tiap generasi atau iterasi dari algoritma.Hanya N_{keep} kromosom yang akan bertahan hidup. $N_{pop} - N_{keep}$ akan dihilangkan sehingga tersedia tempat baru untuk *offspring* baru. Menentukan jumlah

kromosom yang bertahan hidup merupakan sesuatu yang tidak pasti. Jika jumlah kromosom yang bertahan hidup untuk generasi selanjutnya sedikit maka akan membatasi ketersediaan gen-gen pada generasi selanjutnya. Jika sebaliknya, dimana kromosom yang bertahan berjumlah lebih banyak maka ada kemungkinan besar bahwa kromosom-kromosom yang performansinya buruk dapat menurunkan gen-gennya pada generasi selanjutnya. [5]

2.2.3.5 Pemilihan

Tahapan ini merupakan tahapan dimana kromosom-kromosom tersebut akan dipasangkan dan menurunkan 2 kromosom baru. Terdapat beberapa cara pemilihan ini dilakukan: [5]

1. Dipasangkan dari atas ke bawah. Dimulai dengan memasangkan kromosom dari daftar teratas, sampai kromosom N_{keep} teratas telah dipasangkan.
2. Pemilihan acak. Ditunjukkan pada persamaan 2.11, *ceil* merupakan nilai pembulatan ke nilai integer terbesar terdekat.

$$ma = ceil\left(N_{keep} rand(1, N_{keep})\right)$$
$$pa = ceil\left(N_{keep} rand(1, N_{keep})\right) \quad \ldots (2.11)$$

3. Pemilhan acak dengan pembobotan. Terdapat dua teknik dalam melakukan pembobotan yaitu pembobotan berdasarkan pangkat (*rank*) dan pembobotan berdasarkan *cost*.
    a. Pembobotan berdasarkan pangkat ditunjukkan pada Persamaan 2.12.

$$P_n = \frac{N_{keep} - n + 1}{\sum_{n=1}^{N_{keep}} n} \quad \ldots (2.12)$$

    b. Pembobotan berdasarkan *cost* ditunjukkan pada Persamaan 2.13 dan 2.14

$$C_N = C_N - C_{N_{keep+1}} \quad \text{... (2.13)}$$

$$P_n = \frac{C_N}{\sum_m^{N_{keep}} C_m} \quad \text{... (2.14)}$$

4. Pemilihan berdasarkan turnamen. Pemilihan secara acak sebuah subset kecil dari kromosom dan kromosom yang memiliki *cost* terendah akan menjadi orang tua.

2.2.3.6 Perkawinan (*Mating*)

Perkawinan merupakan pembuatan satu atau lebih *offspring* dari orang tua yang dipilih pada proses pemilihan.

2.2.3.7 Mutasi

Mutasi acak akan mempengaruhi persentase tertentu dari bit-bit didalam daftar kromosom. Mutasi dapat menghasilkan sifat yang tidak ada dipopulasi dasar dan menghindari algoritma genetik untuk konvergen terlalu cepat sebelum algoritma genetik men-*sampling* seluruh *cost surface*. Mutasi akan mengubah bit pada kromosom dari 0 menjadi 1, dan juga sebaliknya. [5]

2.2.3.8 Generasi Selanjutnya

Setelah mutasi terjadi, *cost* yang diasosiasikan dengan kromosom *offspring* dan termutasi dihitung. Proses yang dideskripsikan ini diiterasikan. [5]

2.2.3.9 Konvergensi

Jumlah generasi yang berevolusi bergantung dari apakah sebuah solusi yang dapat diterima telah tercapai atau sejumlah iterasi telah terlampaui. Setelah beberapa kali iterasi kromosom dan *cost*-nya dapat menjadi sama jika bukan dikarenakan mutasi. Jika hal ini tercapai maka algoritma sebaiknya dihentikan. [5]

2.3 Pembangunan Perangkat Lunak untuk Solusi menggunakan Microsoft.NET

Microsoft.NET diluncurkan oleh Microsoft dengan berbagai fitur dan komponen yang ditekankan terutama dalam pengembangan *platform*, bahasa, dan protokol. DOT NET ini di dirancang untuk memfasilitasi pengembangan aplikasi web *interoperable* dengan arsitektur yang benar- benar baru. DOT NET ini memungkinkan kita untuk membuat sketsa profil yang relatif akurat dimana mungkin untuk dilihat dalam jangka panjang.

DOT NET adalah *Platform* yang sama sekali baru dengan teknologi yang memperkenalkan sejumlah produk baru, yang tidak menjamin kompatibilitas dengan teknologi yang ada. DOT NET menawarkan dukungan untuk 27 bahasa pemrograman, dimana hirarki kelas dibagi dan menyediakan layanan DOT NET dasar, aplikasi tidak lagi berjalan dalam kode mesin asli dan memiliki kode *abandaned Intel x86* yang mendukung bahasa perantara yang disebut MSIL yang berjalan dalam semacam mesin virtual yang disebut *Common Language Runtime* (CLR).

Arsitektur dari DOT NET sebagai berikut.:

- Merupakan satu set layanan umum yang dapat digunakan dari sejumlah bahasa objek
- Layanan ini dilaksanakan dalam bentuk kode menengah yang independen dari arsitektur yang mendasarinya.
- Mereka beroperasi di runtime (Common Language Runtime) yang mengelola sumber daya dan pelaksanaan aplikasi monitor.

Tujuan utama DOT NET adalah untuk memberikan pengembang dengan sarana untuk membuat aplikasi interoperable menggunakan "Web Services" dari setiap jenis terminal, baik itu PC, PDA, ponsel, dan sebagainya.

Kelebihan DOT NET diantaranya:

1. Multi_Bahasa
2. Aplikasi hardware-independent
3. Aplikasi portabel
4. Bahasanya sesuai dengan kesepakatan bersama

2.4 Perpipaan

2.4.1 Definisi Pipa

Pipa merupakan saluran tertutup, biasanya berupa penampang lingkaran. Pipa bisa terbuat dari berbagai bahan seperti baja atau plastik. Jalur pipa merupakan garis panjang hubungan antara segmen – segmen pipa dan segala sesuatu yang diperlukan untuk operasi sistem. Jalur pipa ini digunakan untuk mengangkut cairan fluida, cairan, padatan, fluid solid, dan lain-lain. [6]

2.4.2 Fungsi Perpipaan

Perpipaan merupakan sarana transportasi. Namun masyarakat kurang memahami karena biasa terdapat didalam bawah tanah sehingga tidak terlihat. Perpipaan sangat penting bagi kesejahteraan ekonomi dan keamanan negara. Perpipaan biasa digunakan untuk komoditas sebagai berikut: [6]

1. Keperluan rumah
2. Aliran air limbah pabrik
3. Transportasi gas alam dari sumber ke konsumen yang jaraknya sangat jauh baik rumah, sekolah, pabrik, atau pembangkit listrik.
4. Transportasi dari ladang minyak mentah ke kilang
5. Mengabungkan Proses permurnian petroleum yang berasal dari berbagai kota

2.4.3 Jenis Perpipaan

Jenis perpipaan bergantung pada komoditas apa yang diangkut. Perpipaan diantaranya ada jaringan pipa air, saluran pembuangan, transportasi gas alam, jaringan pipa minyak mentah, produk olahan minyak bumi, dan juga padatan.

Menurut mekanika fluida pipa diklasifikasikan sebagai aliran satu fase mampat (air, minyak, selokan), aliran satu fasa kompresibel (pipa gas alam), dua fasa campuran padat dan cair *(hydrotransport)*, dua-fase aliran campuran padat-gas (*pneumotransport*), dua aliran campuran fase cair-gas (Minyak-gas pipa), cairan non-Newtonian.

2.4.4 Komponen Perpipaan

Perpipaan merupakan sistem transportasi yang kompleks, ini meliputi komponen- komponen lain diantaranya: [6]

1. Pipa
2. Fiting (katup, kopling, dll)
3. Struktur *inlet* dan *outlet*, pompa (untuk cairan) atau kompresor (untuk gas)
4. Peralatan bantu (*flowmeters*, transduser,perlindungan katodik sistem, dan sistem kontrol otomatis termasuk computer dan pengendali *programable logic*).

2.4.5 Keuntungan dari Perpipaan

Perpipaan sangat dominan untuk proses pengangkutan gas atau cairan maupun padatan dalam jumlah yang besar. Keuntungan dari penggunaan perpipaan adalah:

1. Keekonomian

   Metode pengangkutan cairan, gas, atau padatan dalam jumalah besar serta dataran yang kasar jauh lebih murah daripada pengangkutan dengan truk ataupun kereta api.
2. Penggunaan energi yang rendah

   Intensitas energi pipa jauh lebih rendah dibandingkan dengan truk, dan bahkan lebih rendah dari kereta api. Intensitas energi didefinisikan sebagai energi yang dikonsumsi dalam mengangkut berat kargo atas satuan jarak atau dalam satuan seperti Btu per tonmile.
3. Ramah terhadap lingkungan

   Jaringan pipa sebagian besar dibawah tanah sehingga tidak menyebabkan kebisingan, polusi udara, maupun kemacetan lalu lintas. Pecahnya pipa tentu akan membuat pencemaran terhadap tanah dan lainya namun resikonya lebih kecil dari tumpahan yang mungkin terjadi apabila pengangkutan melalui jalur lainnya.

2.4.6 Jenis Pipa

Kondisi dan aplikasi yang berbeda memerlukan jenis pipa yang berbeda-beda. Untuk memilih pipa apa yang diperlukan untuk sebuah proyek tertentu diperlukan pemahaman karakteristik dari berbagai jenis pipa yang tersedia secara komersial. Ada 2 klasifikasi, logam dan non logam.

2.4.6.1 Pipa Logam

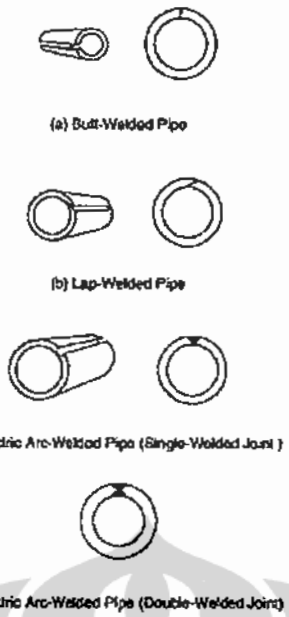
a. Pipa baja biasa

Terbuat dari tempa baja dalam bentuk lembaran yang digulung dan dibentuk lingkaran dengan ujung yang di las. Terdapat 4 jenis pengelasan, yaitu :

1. *Butt –welded Pipe*
2. *Lap-Welded Pipe*
3. *Electric Arc-Welded Pipe (Single-Welded Joint)*
4. *Electric Arc-Welded Pipe (Double-Welded Joint))*

Pipa baja memiliki stuktur yang kuat dan tidak mudah patah namun apabila dilapisi dengan bahan *inert* atau dilindungi dengan bahan lain, makan akan terjadi korosi yang parah.

Pipa baja terdiri dari banyak ukuran dan kekuatan. Biasa digunakan untuk mengangkut gas alam, minyak bumi, udara, dan air. Pipa baja yang digunakan untuk air harus diantisipasi untuk kemungkinan korosi sehingga ada langkah-langkah yang darus dilakukan untuk mengendalikannya. Kekuatan pipa baja berkisar 3000-7000 psi. Untuk tegangannya berdasarkan persentasi dari pipa yang memiliki kekuatan yang bervariasi sesuai kode. Untuk tekanan internal harus berada di bawah 150 psi ketebalan pipa baja ditentukan oleh tekanan eksternal. Sedangkan untuk tekanan internal lebih bersar dari 150 psi.



Gambar 2.4 Gambar Jenis Pipa Logam

b. Pipa baja bergelombang

Pipa baja ini berdiameter besar terbuat dari baja galvanis, lembarannya memiliki satu lipatan heliks atau annular. Karena biaya yang rendah maka pipa baja bergelombang ini digunakan seara luas dalam sistem pembuangan dan *drainase* dimana tekanan dalam dan luarnya rendah. Penggunaan yang demikian tidak akan menjadi masalah yang serius apabila terjadi kebocoran.

c. Pipa besi

Terdapat 2 jenis pipa besi yaitu pipa besi biasa dan pipa besi *ductile*.

1. Pipa besi biasa (abu-abu)

   Pipa besi yang biasa terbuat dari 3-4% karbon untuk serpih grafit. Pipa ini memiliki kekuatan yang relatif kuat dan tahan terhadap korosi, biasa digunakan untuk penyediaan air atau air limbah. Kekurangan pipa ini adalah tidak tahan oleh beban yang berlebihan sehingga mudah pecah.

2. Pipa besi *ductile*

   Pipa ini mengandung 3,5% karbon dan magnesium. Pipa ini lebih lentur dan tidak mudah pecah.

   Penunjukan kekuatan pipa 60-40-10. 60 nilai untuk kekuatan tari minumum sebesar 60.000 psi , 42 untuk nilai kekuatan yield 42.000 psi, 10 untuk nilai perpanjangan pipa minimal 10%, pipa ini iasa digunakan untuk sanitasi *suplay* air dan air limbah.

d. Pipa stainless steel

Pipa ini terbuat dari baja yang berisi paduan krom-nikel. Kelebihan pipa tahan terhadap korosi, namun harga yang mahal menyebabkan pipa ini hanya di gunakan ketika cairan atau lingkungan yang memungkinkan terjadinya korosi, contohnya pada farmasi atau industri makanan.

e. Pipa alumunium

Pipa ini tahan terhadap korosi digunakan untuk tanaman pangan tertentu dan pabrik kimia. Pipa ini bermacam-macam jenisnya dan dapat digunakan sesuai kebutuhan.

f. Pipa tembaga

Pipa ini tahan terhadap korosi namun harganya mahal sehingga pipa ini hanya digunakan utnuk pipa kecil.

g. Pipa logam lainnya

Pipa logam lainnya terbuat dari bahan-bahanyang berbeda utnuk tujuan yang beraneka ragam, seperti tahan terhadap korosi maupununtuk cairan tertentu, dan kodisi-kondisi lainnya.

2.4.6.2 Pipa Non-logam

Pipa non logam tidak sekuat pipa logam namun kelebihan pipa ini adalan ringan dalam berat, juga lebih ekonomis dan taha terhadap korosi.

a. Pipa beton

Pipa beton dibagi menjadi 2 yaitu tekanan rendah dan tekanan tinggi. Tekanan rendah biasanya terbuat dari beton polos biasa digunakan di gorong-gorong yang tidak beroperasi dibawah tekanan internal yang tinggi ataupun yang tekanan yang sedang. Beton polos dapat menahan tekanan eksternal yang tinggi baik oleh panas bumi maupun tekanan lalulintas. Tanpa adanya tulang beton, beton akan mudah retak atau pecah ketika dibawah ketegangan. Pipa beton mampu maahan tekanan internal yang tinggi dengan menempatkan penguat beton dengan menggunakan tulang beton. Biasa digunakan untuk air limbah.

b. Pipa plastic

Terdapat jenis pipa plastik yang berada di pasaran umumnya adalah PVC (*polyvinyl chloride*), PE(*Polyethylene*), dan PP (*polypropylene*).

Pipa ini biasa digunakan untuk air, air limbah, gas alam, dan cairan bertentu yang tidak bereaksi dengan bahan plastik. Kelebihan dari pipa jenis ini adalah murah, ringan, tidak korosi, namaun sayang nya pipa ini tidak sekuat logam, mudah berubah bentuk, rapuh pada cuaca yang dingin, dan dapat meluas akibat suhu tinggi

c. Pipa keramik (*clay*)

Biasa digunakan di selokan, murah, tahan terhadap korosi yang tinggi. Memiliki kekurangan yaitu mudah rapuh, tidak tahan pada tekanan internal.

d. Pipa kayu dan bamboo

Pipa jenis ini biasa digunakan untuk transmisi air. Banyak digunakan secara luas di asia timur terutama cina untuk mengangkut air. Biaya rendah dan ramah lingkungan.

e. Pipa *carbon* grafit

Mereka sangat rapuh, dan hanya digunakan dalam aplikasi khusus yang melibatkan suhu yang sangat tinggi.

f. Pipa semen

Jenis in biasa di gunakan untuk saluran pembuangan dan saluran air. Digunakan di hampir seluruh negara, kecuali Amerika karena diketahui menyebabkan kanker paru-paru karena menghirup asbestos

g. Pipa karet

Karet alam dan karet sintetis digunakan untuk membuat selang yang fleksibel. Pipa ini dibutuhkan untuk skala kecil dan aplikasi khusus seperti berkebun dan mesin hidrolik

h. Pipa kaca

Kelemahan pipa jenis ini adalah rapuh dan mahal, namun sangat aman untuk cairan yang bersifat korosif dantahan terhadap suhu yang tinggi. Biasa digunakan di pabrik makanan, farmasi, dan pabrik kimia karena tahan terhadap zat yang asam

i. Tubing

Pipa kecil dengan dinding yang tipis. Fitting dan katup untuk pipa tidak dapat digunakan untuk tubing tanpa modifikasi. Tubing terbuat dari baja, besi, stainless steel, aluminium, tembaga, dan plastik. Pipa ini dibuat untuk digunakan dalam pengeboran dan perawatan sumur minyak dan gas, dan pemrosesan makanan, atau pabrik farmasi

2.4.7 Komponen perpipaan

Pipa memiliki komponen seperti panjang, ketebalan , dan diameter. Diameter ada 2 yaitu diameter (*schedule*) dalam dan luar. Ketebalan pipa dikelompokan sebagai berikut:

1. *Schedule* : 5,10,20,30,40,60,80,100,120,160
2. *Schedule standard*
3. *Schedule ectra strong (XS)*
4. *Schedule double extra strong (XXS)*
5. *Schedule special*

Perbedaan *schedule* dibuat untuk :

1. Menahan internal pressure dari aliran
2. Kekuatan dari material
3. Mengatasi karat
4. Mengatasi kegetasan pipa

2.4.8 Hubungan antara laju alir dan kehilangan tekanan

Tekanan yang ada dalam pipa dapat menyebabkan kebocoran. Maka tekanan yang ada di dalam pipa harus dibatasi dengan suatu nilai tertentu. Dimana tekanan maksimum berada pada batas toleransi kekuatan pipa, hal ini bergantung pada spesifikasi para produsen pipa. Maka solusi utnuk mengatasi masalah ini adalah harus mengkondisikan tekanan gas yang mengalir didalam pipa harus memenuhi persamaan laju alir gas.

Kehilangan tekanan (Δp) dari aliran gas melalui suatu pipa bergantung dari laju alis gas, panjang pipa serta diameter pipa. Untuk aliran gas alam dengan tekanan yang maksimal 7 bar, dapat didekati dengan menggunakan persamaan Panhandle A yang ditunjukkan pada Persamaan 2.15

$$\Delta P = P_1{}^2 - P_2{}^2 = KQ^{1.8554} \quad \text{... (2.15)}$$

Dimana,

$$K = 19.43\left(\frac{L}{D^{4.854}E^2}\right) \quad \text{... (2.16)}$$

Keterangan :

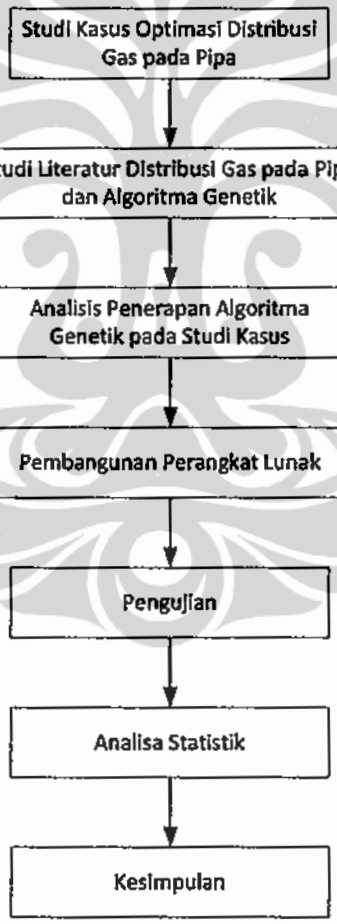
D = diameter pipa (mm)
L = panjang pipa (m)
P_1 = tekanan *inlet* (bar)
P_2 = tekanan *outlet* (bar)
Q = laju alir gas (m^3/h)
E = efisiensi faktor (0.9)

BAB 3

METODE PENELITIAN

3.1 Metodologi

Dari latar belakang, perumusan masalah, tujuan penelitian dan batasan masalah yang telah diuraikan pada bab sebelumnya, maka dapat disusun metode dan langkah kerja yang akan dilakukan dalam penelitian ini. Skema prosedur penelitian yang akan dilakukan dapat dilihat pada Gambar 3.1. dengan tahapan sebagaimana di bawah ini.



Gambar 3.1 Skema Prosedur Penelitian Optimasi Jaringan Pipa untuk Distribusi Gas menggunakan Algoritma Genetik

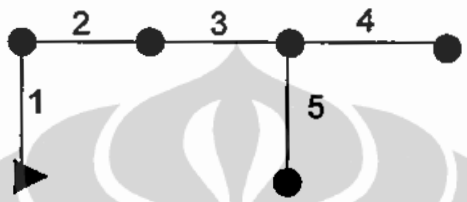
Uraian lebih detail mengenai masing- masing tahapan di atas alam dibahas pada sub bab berikutnya.

3.2 Pemodelan Jaringan Pipa Gas

Jaringan pipa akan dimodelkan seperti Gambar 3.2. Lambang-lambang yang digunakan untuk membantu memodelkan yaitu:

- ● = *Node*
- ▶ = Sumber *pressure*

Pada Gambar 3.2, terdapat 5 *node*, 5 pipa dan 1 sumber *pressure*.



Gambar 3.2 Contoh Pemodelan Jaringan Pipa untuk Distribusi Gas

3.3 Formulasi *cost* optimum pada jaringan perpipaan

Tujuan dari studi kasus ini adalah untuk menimalisasikan biaya yang dibutuhkan untuk membangun sebuah jaringan pipa gas, berdasarkan data-data yang dimiliki mengenai diameter pipa dan biayanya per satuan panjang. Secara garis besar solusi untuk permasalahan yang kita hadapi ditunjukkan pada Persamaan 3.4

$$minimal\ f(x) = Cost(d) \quad \text{... (3.4)}$$

Keterangan:

d = Diameter pipa

Fungsi *cost* sendiri ditunjukkan pada Persamaan 3.5

$$Cost(d) = \sum_{j=1}^{n} l_j c(d_j) \quad \text{... (3.5)}$$

Keterangan:

l_j = Panjang dari pipa (j)

$c(x)$ = Harga per satuan panjang dari diameter x

d_j = Diameter pipa (j)

Terdapat beberapa kondisi (*constraint*) yang harus dipenuhi dalam sebuah jaringan pipa gas, yaitu:

- Berdasarkan hukum Kirchoff yang pertama, dikatakan bahwa dalam tiap *node*, gas yang masuk harus sama dengan gas yang keluar. Hukum ini dapat dirumuskan seperti Persamaan 3.6

$$Q_j = \sum_{i=1}^{n} a_{ij} q_i, j = 1,2,\ldots,m \quad \ldots (3.6)$$

Keterangan:

Q = Kebutuhan gas (j)

a_{ij} = $\begin{cases} -1 \\ +1 \\ 0 \end{cases}$

Bernilai -1 jika gas masuk kedalam *node*. +1 jika gas keluar dari *node*, 0 jika tidak ada koneksi pipa antara *node i* dan *j*

q_i = Aliran Gas pada pipa (j)

3.4 Analisis Penerapan Algoritma Genetik pada Studi Kasus

Berikut akan dijabarkan secara detail bagaimana algoritma genetik bekerja pada kasus optimalisasi diameter pipa. Penjabaran ini berdasarkan pada teori algoritma genetik pada Bab 2 Gambar 2.2.

Pada Gambar 3.2 akan dipasangkan pipa secara acak. Misalkan pemasangan pipa acak ini menghasilkan pasangan pipa sebagaimana pada tabel 3.1. Pasangan ini dianggap sebagai merupakan solusi (parent A).

Tabel 3.1 Iterasi Awal Solusi Optimasi Jaringan Pipa

Jalur	Pemasangan pipa secara acak Diameter pipa (mm)	Kode biner	Biaya/*path* ($)
1	90	010	884000
2	32	000	350000
3	125	011	1575000
4	180	100	2662500
5	180 (high density)	110	3000000

Kode parent A menjadi: 010 000 011 100 110 dengan biaya : US$ 8471500. Pengacakan solusi ini dibuat 4 kali sehingga menghasilkan 4 parent yang *P* minimumnya tiap *node* terpenuhi, Dari 4 parent yang sudah di bentuk akan diseleksi 2 terbaik berdasarkan *cost* paling minimum.

Dari 2 parent terbaik akan dilakukan pekawinan sesuai dengan teori yang ada pada bab 2 Gambar 2.2 dilakukan terus sampai mendapatkan nilai *cost* yang minimum dengan aturan P minimum dan Q tiap *node* yang terpenuhi.

3.5 Pembangunan Perangkat Lunak

Workflow pada Perangkat Lunak yang akan dibangun ditunjukkan pada Gambar 3.4.

Gambar 3.3 *Workflow* Perangkat Lunak Optimasi Jaringan Pipa

3.5.1 *Input*

Beberapa input yang perlu diberikan kepada sistem yaitu:

- Tabel jenis diameter dan harganya per satuan panjang

  Dikarenakan Algoritma Genetik hanya dapat memproses biner maka akan dilakukan pengkodean (*encoding*) pada jenis diameter. Contohnya dapat dilihat pada Tabel 3.2

Tabel 3.2 Contoh *input* diameter dan harganya

Diameter pipa (mm)	Harga ($ / m)		Index	Kode biner
32	100	→	1	000
63	140	→	2	001
90	170	→	3	010
125	250	→	4	011
180	375	→	5	100
250	580	→	6	101

- Banyaknya *node* dengan data Q dan P minimum

- Jarak-jarak pada Gambar 3.2 ditunjukkan pada Tabel 3.3.

Tabel 3.3 Contoh *input* diameter dan harganya

Jalur	Jarak (m)
1	5200
2	3500
3	6300
4	7100
5	5000

3.5.2 *Output*

Keluaran dari sistem akan berupa presure yang diterima oleh tiap *node* dengan Q yang sesuai dengan permintaan dan *cost* minimum yang diperoleh dari hasil Algoritma Genetik.

3.6 Pengujian

Pada tahapan ini pembangunan perangkat lunak telah dilakukan lalu dilakukan pengecekan validasi dari seluruh sistem. Simulasi dapat dilakukan apabila program sudah berjalan dengan baik dan segala formulasi telah tepat.

3.7 Analisa Statistik

Simulasi dilakukan berkali – kali sehingga ada perbandingan hasil simulasi. Dari data statistik akan terlihat apakah kondisi optimum diperoleh untuk mengoptimasi *cost* dari jaringan perpipaan.

3.8 Kesimpulan

Hasil dari simulasi dan analisa statistik akan memberikan kesimpulan akhir mengenai optimasi jaringan perpipaan pada suatu contoh kasus.

BAB 4

HASIL SIMULASI PROGRAM

Sebelum membahas mengenai hasil dari simulasi akan dijelaskan mengenai deskripsi singkat mengenai program yang dibangun, laju alur program dan *user manual*.

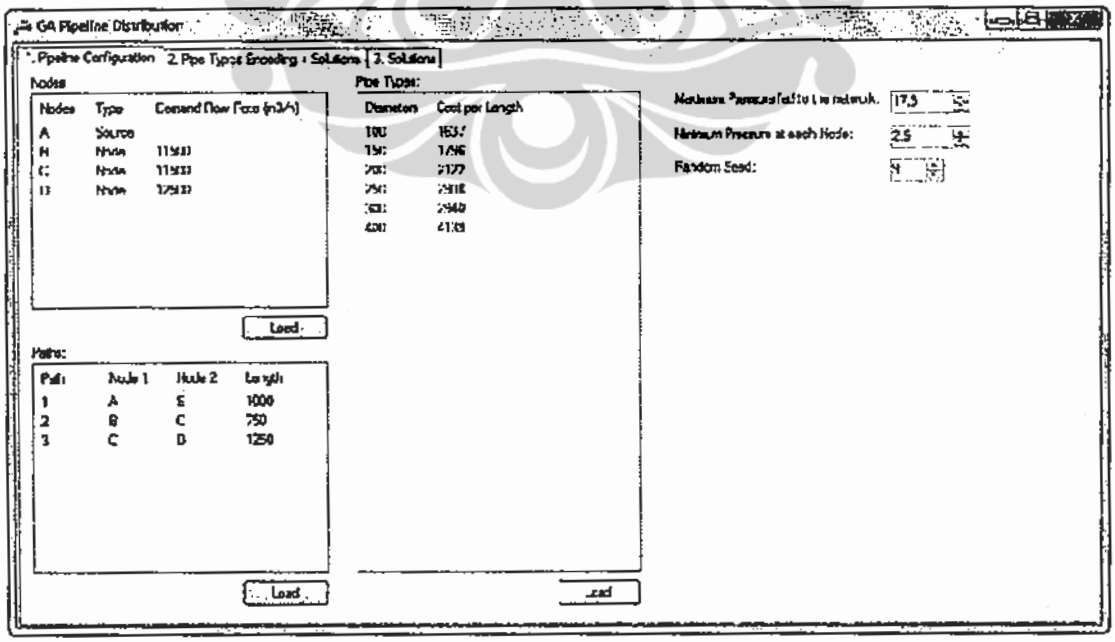
4.1 Tampilan Program dan User Manual

Interface program yang dibangun dibagi menjadi 3 bagian, yaitu:

1. Pipeline Configuration (Konfigurasi Jaringan Pipa)
2. Pipe Type Encoding & Solution (Pengkodean Tipe Pipa dan Solusi)
3. Final Solution (Solusi Akhir)

4.1.1 Konfigurasi Jaringan Pipa

Tampilan untuk bagian konfigurasi jaringan pipa dapat dilihat pada Gambar 4.1. Tampilan ini mengatur segala aturan jaringan pipa gas.



Gambar 4.1 Tampilan Konfigurasi Jaringan Pipa

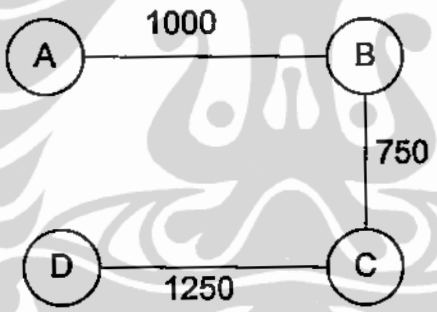
Dalam bagian Pipeline Configuration dibagi menjadi 4 bagian, yaitu:

1. *Node*, disini *user* dapat *load* data dalam format *notepad* dengan format berurutan dimana baris pertama merupakan *node source* dibatasi dengan "tab" tipe dari setiap *node* (*source or node*), lalu dibatasi dengan "tab" lalu besar *Demand Flow Rate* dengan satuan m^3/h.

   Contoh isi dari format *notepad* untuk input *node*:

A	Source	0
B	Node	11500
C	Node	11500
D	Node	12500

2. *Path*, disini *user* dapat *load* data dalam format *notepad* dengan format berurutan berdasarkan skema jaringan dengan urutan *node* yang berhubungan dengan dibatasi oleh "tab", contoh dapat dilihat pada



Gambar 4.2 Contoh Path

   Maka format *notepad* nya menjadi:

1	A	B	1000
2	B	C	750
3	C	D	1250

3. *Pipe Type*, pada bagian ini *user* dapat load data diameter pipa yang ada dipasaran (mm) dan harganya dengan dibatasi "tab", contohnya sebagai berikut:

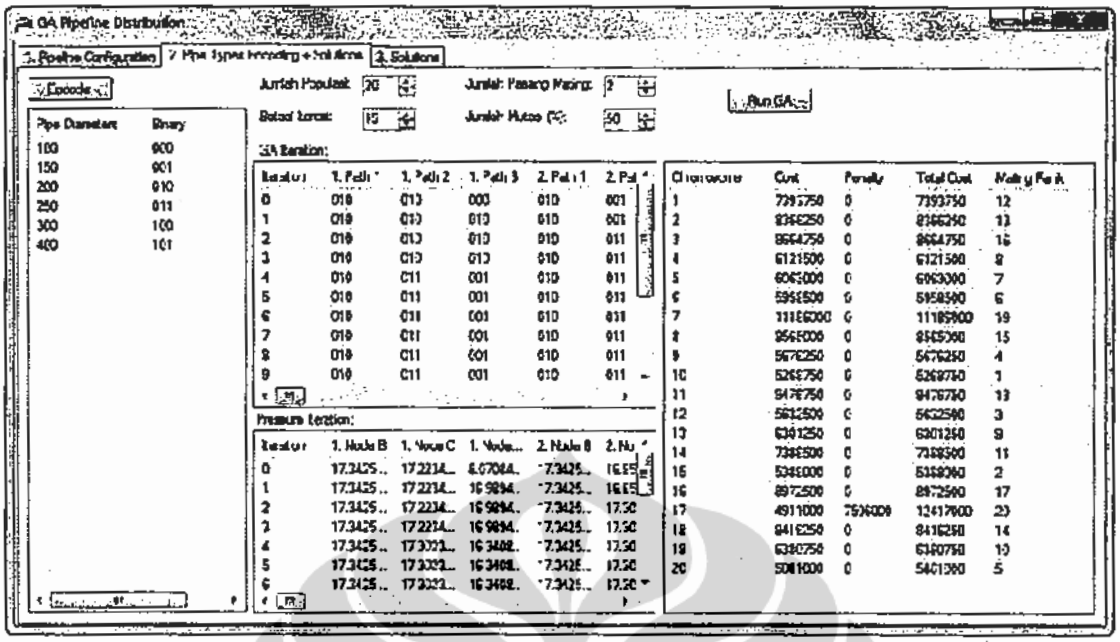
32	100
63	140
90	170
125	250
180	375
250	580

4. Data tambahan diantaranya *user* harus memasukan,diantaranya:
   - *maximum Pressure Fed to the* network, nilai ini adalah besaran *pressure* yang masuk dalam jaringan yang berasal dari *node sourde*
   - *Minimum Pressure at each Node,* nilai ini sebagai batasan dari minimum *pressure* yang diterima dari setiap *node*. Jika nilai *pressure* yang diterima dibawah minimum *pressure* maka hasil belum valid
   - *Random seed* untuk 'mengunci' angka acak.

4.1.2 Pengkodean Tipe Pipa dan Solusi

Gambar 4.3 adalah tampilan untuk konfigurasi dan data hasil Algoritma Genetik, ditampilan ini dengan "klik" *encode* maka program akan membuat *code binary* untuk tiap ukuran diameter yang sebelumnya di input pada tampilan Pipeline Configuration. Input berikutnya yang harus dimasukan adalah:

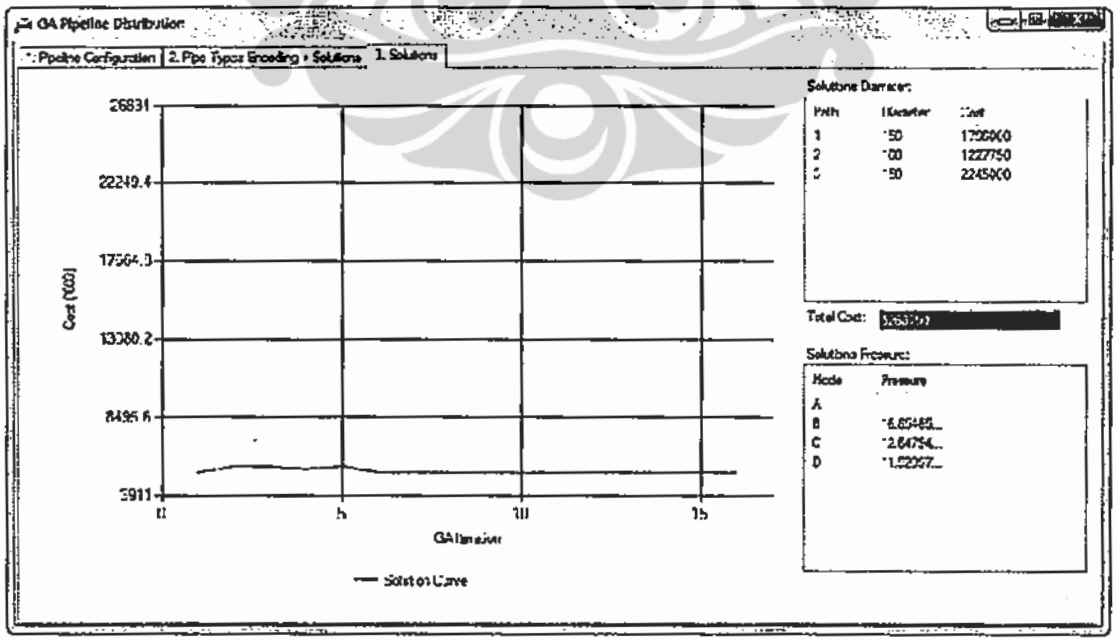
- Jumlah Populasi, nilai ini untuk menentukan jumlah komosom
- Batas Interasi, nilai ini adalah untuk membatasi iterasi dimana jika iterasi sudah sampai ke batas iterasi maka iterasi akan berhenti
- Jumlah Pasang Mating, adalah jumlah pasangan yang akan dikawinkan untuk menghasilkan populasi baru.
- Jumlah Mutasi, adalah persen dari jumlah populasi yang akan terkena mutasi.



Gambar 4.3 Tampilan Pengkodean Tipe Pipa dan Solusi

4.1.3 Solusi Akhir

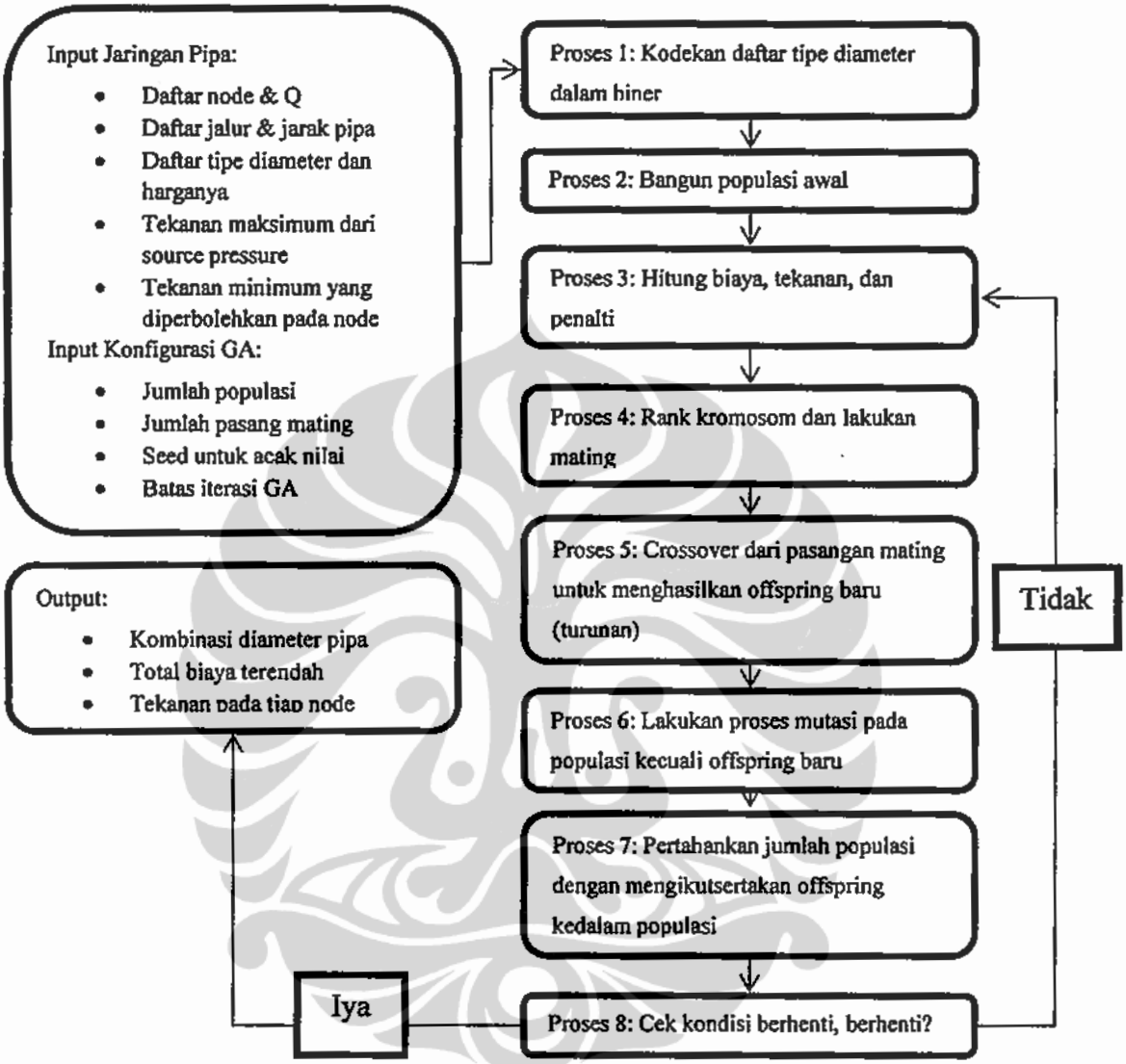
Gambar 4.4 adalah tampilan solusi diantaranya grafik *cost* yang dihasilkan dari program Algoritma Genetik, ukuran diameter untuk setiap *path*, *cost* terendah yang telah diperoleh, dan *pressure* untuk setiap node.



Gambar 4.4 Tampilan Solusi Akhir

4.1.4 Laju Alur Program

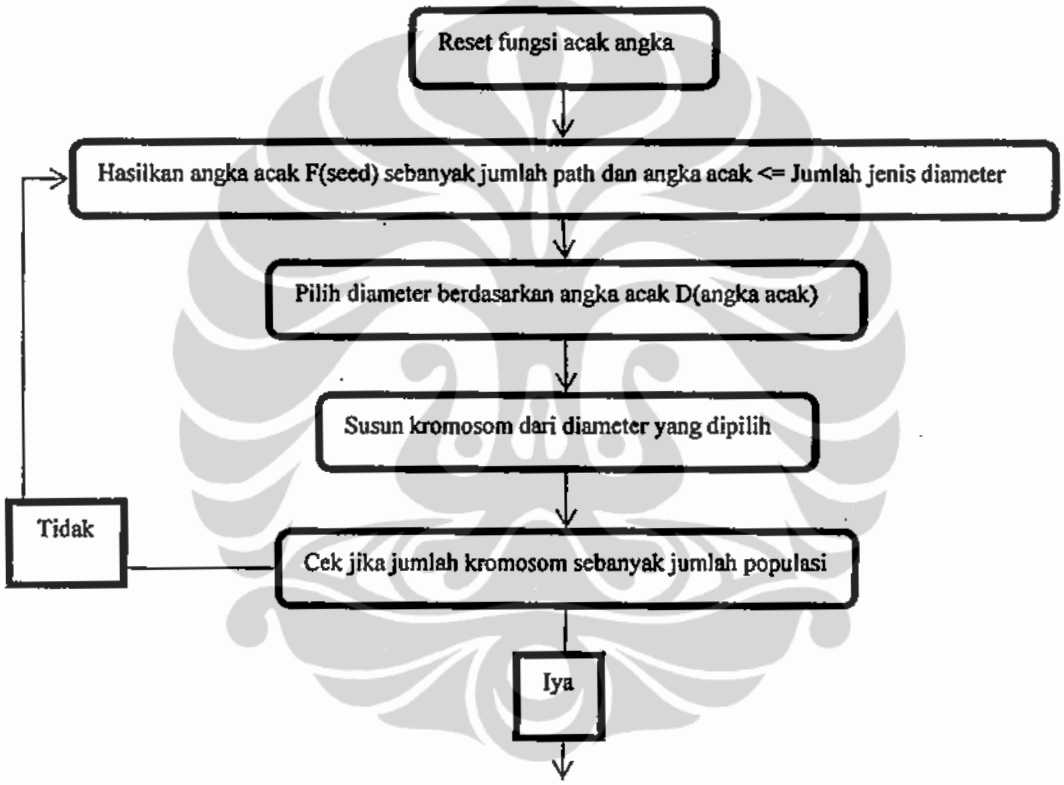
Gambar 4.5 menggambarkan alur dan cara kerja dari program secara garis besar.



Gambar 4.5 Diagram Alur Program

Program dimulai dengan memasukan input, kemudian untuk input daftar diameter pipa dikodekan menjadi biner, misalnya untuk diameter pipa 100 mm dikodekan menjadi 01. Langkah selanjutnya yaitu membangun populasi. Populasi sendiri dibangun atas kromosom-kromosom, tiap kromosom ini akan merepresentasikan sebuah solusi dari permasalahan pipa. Misalkan terdapat 3 jalur pipa yang akan kita optimal biayanya, maka sebuah kromosom dapat

merepresentasikan kombinasi dari diameter pipa seperti 01 01 01. Pada proses ini populasi akan dibangun secara acak, nilai acaknya ini tergantung dari nilai seed yang digunakan. Seed dapat dianalogikan sebagai sebuah jalur yang berisi angka acak. Selama kita menggunakan seed yang sama maka nilai acak yang dihasilkan akan selalu sama. Misalnya seed 9 akan menghasilkan angka acak secara berurutan 3, 4, 5, 1, 9, 2, dan seterusnya. Maka jika dijalankan lagi dengan nilai seed yang sama, maka akan menghasilkan angka acak yang sama dengan urutan yang sama. Berikut ini merupakan alur bagaimana populasi diciptakan.



Gambar 4.6 Diagram Proses Populasi Awal Dibuat

Dari kromosom-kromosom yang dihasilkan, hitung biaya dan tekanan dari tiap node. Biaya diperoleh dengan mengkalikan harga diameter per jarak dengan jarak pipanya. Tekanan ditiap node diperoleh melalui rumus PanHandle A. Kemudian cek constraint, jika terdapat sebuah node yang melanggar maka diberikan penalti.

Langkah berikutnya adalah rank kromosom pada populasi berdasarkan biaya dan penalti. Setelah di rank maka dilakukan proses mating dimana perkawinan dilakukan antar kromosom diurutkan berdasarkan rank pertama. Jumlah mating yang dilakukan adalah berdasarkan input yang diberikan diawal.

Setelah pasangan mating dipilih, crossover kemudian dilakukan. Crossover akan menghasilkan kromosom baru dengan gen-gen yang merupakan kombinasi dari kedua orangtuanya. Crossover ini akan menghasilkan dua offspring atau anak.

Berikutnya melibatkan mutasi dari populasi yang ada sekarang. Mutasi merupakan proses dimana gen pada sebuah kromosom berubah secara acak. Mutasi penting dilakukan karena akan memperbanyak kemungkinan solusi.

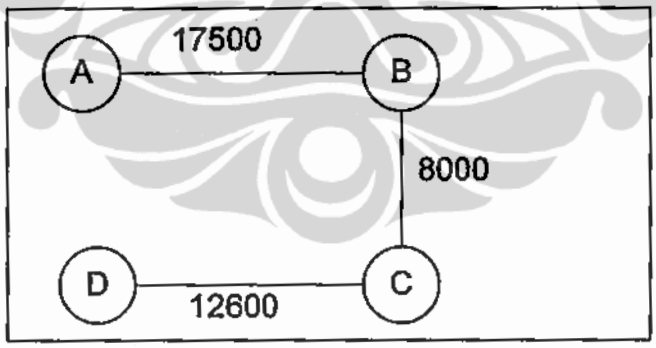
Setelah mutasi dilakukan, offspring baru dimasukan kedalam populasi dan kromosom yang lama dihilangkan. Jumlah populasi akan dipertahankan untuk menghemat resource komputasi. Lalu pengecekan dilakukan jika kondisi berhenti telah tercapai, maka kromosom dengan rank kesatu dikeluarkan sebagai hasil solusi dari algoritma genetik untuk permasalahan ini.

4.2 Validasi Hasil antara Perhitungan Program dan Perhitungan Manual untuk Kasus Sederhana

Gambar 4.7, Tabel 4.2, Tabel 4.3, Tabel 4.4, dan Tabel 4.5 adalah data yang dijadikan contoh simulasi untuk hitungan manual untuk dibandingkan dengan hasil program. Cuplikan data harga yang digunakan, diambil dari contoh kasus pada jurnal *Computer aided optimization of natural gas pipe networks using genetic algoritm* oleh Omar Fayez (Mei 2008) yang dapat dilihat pada Tabel 4.1.

Tabel 4.1 Daftar Diameter beserta Price dari Jurnal

Diameter (mm)	Harga ($/m)
100	1650
150	1790
200	2122
200 (*high density*)	2950
250	2908
250	4000
300	2940
350	3069
400	4139
450	4450
500	4690
550	4952
600	5012
650	5114
700	5200



Gambar 4.7 Kasus 1 – Diagram Jaringan Pipa

Tabel 4.2 Kasus 1 – Daftar Node

Node	*Demand Flow Rate* (m^3/h)
A	0
B	11500
C	8700
D	11500

Tabel 4.3 Kasus 1 – Daftar Jalur Pipa

Node Asal	Node Tujuan	Jarak (m)
A	B	17500
B	C	8000
C	D	12600

Tabel 4.4 Kasus 1 – Daftar Diameter Pipa

Diameter pipa (mm)	Harga ($/m)
100	1650
150	1790
200	2950
250	4000

Tabel 4.5 Kasus 1 – Tekanan dari Source dan Tekanan Minimum ditiap Node

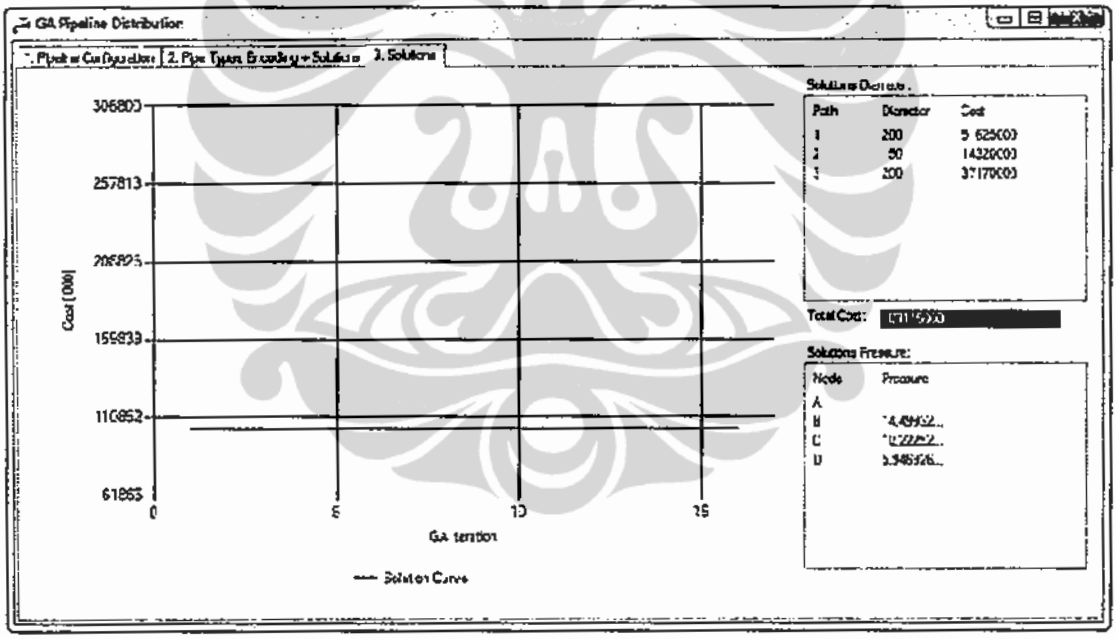
Pressure fed to the network **(bar)**	17,5
Minimum pressure at each Node **(bar)**	2,5

Dengan mengkombinasikan seluruh ukuran diameter pipa yang tersedia dalam skema jaringan, maka hasil dari simulasi dengan perhitungan manual yang di peroleh dapat dilihat pada Tabel 4.6. Untuk data yang detail dapat dilihat pada Lampiran 1.

Tabel 4.6 Kasus 1 – Hasil Solusi

Node Asal	Node Tujuan	Ukuran Diameter pipa yang cocok (mm)	*Pressure* yang di peroleh (bar)	Jarak (m)	Harga *path* ($)
A	B	200	14,49932	17500	51625000
B	C	150	10,22253	8000	14320000
C	D	200	5,946926	12600	37170000
				TOTAL	103115000

Hasil dengan data yang sama dengan menggunakan program Algoritma Genetik dapat dilihat di Gambar 4.8. Dengan parameter jumlah populasi 20, batas iterasi 15, dan jumlah pasang mating 4.

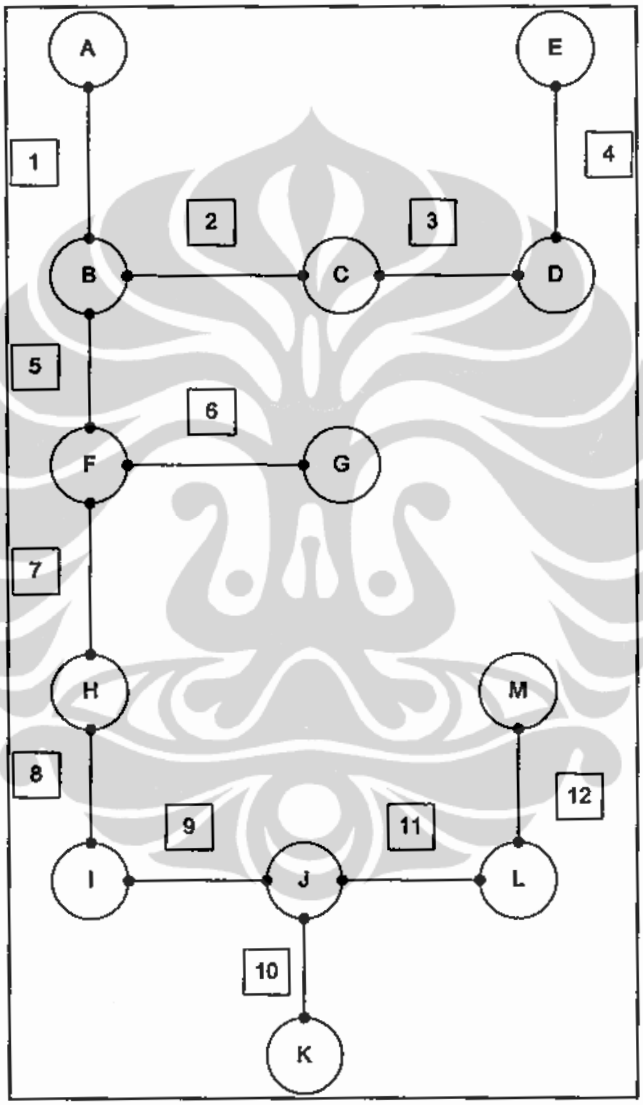


Gambar 4.8 Kasus 1 – Hasil Akhir Menggunakan Program

Dengan hasil dari perhitungan manual dan perhitungan program menghasilkan hasil yang sama, maka perhitungan program dianggap *valid* dan dapat digunakan untuk perhitungan skema jaringan yang lebih rumit.

4.3 Simulasi Program dengan Skema Jaringan yang Lebih Kompleks

Program yang dibangun mampu menghasilkan solusi untuk kasus jaringan yang lebih kompleks dengan tetap pada batasan yang telah diuraikan dalam batasan masalah pada Bab 1. Gambar 4.9 merupakan suatu contoh kasus skema jaringan yang lebih kompleks. Tabel 4.7, Tabel 4.8, Tabel 4.9, dan Tabel 4.10 merupakan data untuk inputan pada program.



Gambar 4.9 Kasus 2 – Diagram Jaringan Pipa

Tabel 4.7 Kasus 2 – Daftar Node

No	*Node*	*Demand Flow rate* (m^3/h)
1	A	0
2	B	8700
3	C	11500
4	D	12500
5	E	11000
6	F	8000
7	G	7500
8	H	11000
9	I	12500
10	J	8700
11	K	8700
12	L	8000
13	M	11500

Tabel 4.8 Kasus 2 – Daftar Jalur Pipa

Path	Jarak (m)
1	8400
2	9200
3	1400
4	8400
5	1700
6	10400
7	12300
8	3540
9	4900
10	15700
11	6300
12	8600

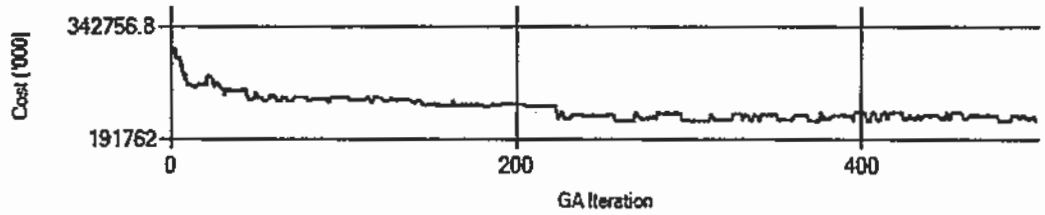
Tabel 4.9 Kasus 2 – Daftar Diameter Pipa

Diameter (mm)	Harga ($/m)
200	2122
250	2908
300	2940
350	3069
400	4139
450	4450
500	4690
550	4952
600	5012
650	5114
700	5200

Tabel 4.10 Kasus 2 – Tekanan dari Source dan Tekanan Minimum ditiap Node

Pressure fed to the network **(bar)**	17,5
Minimum pressure at each Node **(bar)**	2,5

Dari *running* program hasil yang diperoleh grafik seperti yang tergambar pada Gambar 4.10.



Gambar 4.10 Kasus 2 – Hasil Akhir Menggunakan Program

Dilihat dari grafik maka pencarian solusi oleh program menunjukan penurunan untuk angka total *price*, hal ini memperlihatkan bahwa hasil yang diperoleh untuk beberapa iterasi yang dlakukan mengalami penurunan sehingga hasil akhir yang dihasilkan adalah nilai minumum untuk skema jaringan pada Gambar 4.9. Pada Tabel 4.11 telah diurutkan untuk setiap *path* ukuran pipa berapa saja yang dinilai optimum secara keseluruhan untuk jaringan pada Gambar 4.9. Pressure yang diterima untuk setiap *path* diperlihatkan pada Tabel 4.12.

Tabel 4.11 Kasus 2 – Hasil Akhir Diameter Pipa yang Digunakan

Path	Diameter (mm)	Harga ($/m)	Harga/*path* ($)
1	200	2.282	17.824.800
2	200	2.250	19.522.400
3	200	2.378	2.970.800
4	200	2.186	17.824.800
5	300	2.154	4.998.000
6	200	2.122	22.068.800
7	200	2.314	26.100.660
8	200	2.186	7.511.880
9	200	2.378	10.397.800
10	200	2.154	33.315.400
11	550	2.314	31.197.600
12	300	2.218	25.284.000
		TOTAL	21.9016.880

Tabel 4.12 Kasus 2 – Hasil Akhir Tekanan pada Jaringan Pipa untuk tiap node

Node	*Pressure* yang diterima (bar)
B	16,6966
C	15,1095
D	14,4098
E	13,2999
F	16,6766

Tabel 4.12 Kasus 2 – Hasil Akhir Tekanan pada Jaringan Pipa untuk tiap node (Lanjutan)

Node	*Pressure* yang diterima (bar)
G	16,6671
H	14,6956
I	13,9029
J	13,3140
K	11,2210
L	13,3092
M	13,0591

4.4 Penelitian Simulasi dengan Perubahan Parameter

Pada sub-bab ini akan dijelaskan diteliti parameter-parameter pada algoritma genetik dalam perannya untuk menemukan sebuah solusi.

Studi kasus 3 yang digunakan yaitu digambarkan pada Gambar 4.9, kebutuhan ditiap node yaitu dapat dilihat pada Tabel 4.13, dengan minimal pressurenya yaitu 2,5 bar. Dengan pressure yang dikasih kejaringan tersebut adalah 17,5 bar. Untuk jenis diameter yang tersedia, dapat dilihat pada Tabel

Tabel 4.13 Kasus 2 – Daftar Diameter Pipa

Diameter pipa (mm)	Harga ($/m)
100	1637
150	1796
200	2122
250	2908
300	2940
400	4139

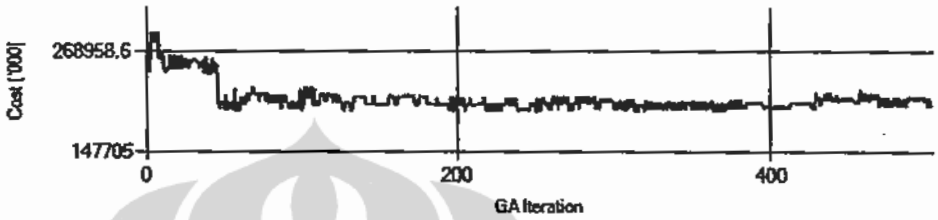
4.4.1 Pengaruh Jumlah Populasi dalam Kontribusinya terhadap Solusi

Parameter kunci yang digunakan yaitu:

- Batas Iterasi: 500
- Jumlah Pasang Mating: 3
- Jumlah Mutasi : 50 %

Hasil penelitian:

- Jumlah Populasi = 10
  Solusi:
  - Biaya = 205.119.660
  - Minimum Pressure = 11,95 bar
  - Grafik iterasi GA dalam mencari solusi ditunjukkan pada Gambar 4.11.



Gambar 4.11 Penelitian 1, Jumlah Populasi = 10

  - Diameter yang digunakan pada tiap path ditunjukkan pada Tabel 4.14.

Tabel 4.14 Hasil Diameter Penelitian 1, Jumlah Populasi = 10

Path	Diameter (mm)	Harga/*path* ($)
1	200	17824800
2	200	19522400
3	200	2970800
4	200	17824800
5	150	3053200
6	150	18678400
7	200	26100600
8	400	14652060
9	300	14406000
10	200	33315400
11	300	18522000
12	200	18249200
	Total	205119660

Node	*Pressure* (bar)
A	
B	16,69
C	15,1
D	14,8
E	13,29
F	16,11
G	16,07
H	14,04
I	14,02
J	13,94
K	11,95
L	13,85
M	12,02
Min	11,95

- Jumlah Populasi = 20
  Solusi:
  - Biaya = 198.627.780
  - Minimum Pressure = 5,57 bar
  - Grafik iterasi GA dalam mencari solusi ditunjukkan pada Gambar 4.12.

Gambar 4.12 Penelitian 1, Jumlah Populasi = 20

- Diameter yang digunakan pada tiap path ditunjukkan pada Tabel 4.15.

Tabel 4.15 Hasil Diameter Penelitian 1, Jumlah Populasi = 20

Path	Diameter (mm)	Harga/*path* ($)
1	200	17824800
2	150	16523200
3	300	4116000
4	200	17824800
5	200	3607400
6	150	18678400
7	200	26100600
8	200	7511880
9	150	8800400
10	200	33315400
11	400	26075700
12	200	18249200
	Total	198627780

Node	*Pressure* (bar)
A	
B	16,69
C	8,64
D	8,57
E	5,57
F	16,55
G	16,51
H	14,55
I	13,75
J	11,15
K	8,54
L	11,12
M	8,75
Min	5,57

- Jumlah Populasi = 30
  Solusi:
  - Biaya = 195.795.480
  - Minimum Pressure = 5,73 bar
  - Grafik iterasi GA dalam mencari solusi ditunjukkan pada Gambar 4.13.

Gambar 4.13 Penelitian 1, Jumlah Populasi = 30

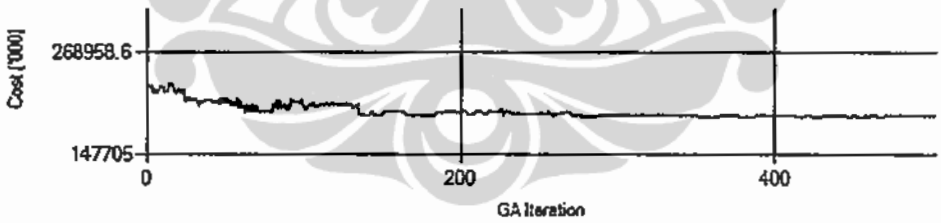
- Diameter yang digunakan pada tiap path ditunjukkan pada Tabel 4.16.

Tabel 4.16 Hasil Diameter Penelitian 1, Jumlah Populasi = 30

Path	Diameter (mm)	Harga/*path* ($)
1	200	17824800
2	200	19522400
3	200	2970800
4	200	17824800
5	250	4943600
6	100	17024800
7	300	36162000
8	200	7511880
9	250	14249200
10	150	28197200
11	150	11314800
12	200	18249200
	Total	195795480

Node	*Pressure* (bar)
A	
B	16,69
C	15,1
D	14,8
E	13,29
F	16,64
G	16,36
H	16,38
I	15,67
J	15,5
K	5,73
L	13
M	11,04
Min	5,73

- Jumlah Populasi = 100
  Solusi:
  - Biaya = 195.166.480
  - Minimum Pressure = 4,53 bar
  - Grafik iterasi GA dalam mencari solusi ditunjukkan pada Gambar 4.14.



Gambar 4.14 Penelitian 1, Jumlah Populasi = 100

  - Diameter yang digunakan pada tiap path ditunjukkan pada Tabel 4.17.

Tabel 4.17 Hasil Diameter Penelitian 1, Jumlah Populasi = 100

Path	Diameter (mm)	Harga/*path* ($)
1	200	17824800
2	200	19522400
3	150	2514400
4	150	15086400
5	200	3607400
6	150	18678400
7	200	26100600
8	200	7511880
9	300	14406000
10	200	33315400
11	150	11314800
12	300	25284000
	Total	195166480

Node	*Pressure* (bar)
A	
B	16,69
C	15,1
D	13,85
E	4,53
F	16,55
G	16,51
H	14,55
I	13,75
J	13,67
K	11,64
L	10,75
M	10,44
Min	4,53

Kesimpulan dari perubahan parameter populasi adalah:

Jumlah populasi mempengaruhi banyaknya kombinasi solusi yang tersedia, semakin banyak populasinya maka semakin besar kemungkinannya algoritma genetik menemukan solusi paling optimal. Semakin banyak populasi maka komputasi pun semakin bertambah dan diperlukan iterasi Algoritma Genetik yang lebih banyak sehingga dapat mencoba segala kemungkinan kombinasi solusi yang ada. Banyaknya populasi akan mempengaruhi total optimasi *cost* yang dihasilkan, semakin besar populasi semakin menunjukan harga optimum cost yang lebih baik. Sedangkan untuk nilai *pressure* semakin banyak populasi tidak mempengaruhi *pressure* minimun di node dalam jaringan

4.4.2 Pengaruh Jumlah Pasang Mating dalam Kontribusinya terhadap Solusi

Parameter kunci yang digunakan yaitu:

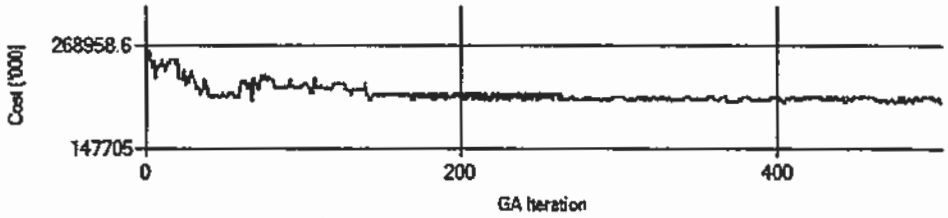
- Jumlah Populasi: 30
- Batas Iterasi: 500
- Jumlah Mutasi: 50 %

Hasil penelitian:

- Jumlah Pasang Mating = 2

Solusi:

- Biaya = 203.484.000
- Minimum Pressure = 4,83 bar
- Grafik iterasi GA dalam mencari solusi ditunjukkan pada Gambar 4.15.



Gambar 4.15 Penelitian 2, Jumlah Pasang Mating = 2

- Diameter yang digunakan pada tiap path ditunjukkan pada Tabel 4.18.

Tabel 4.18 Hasil Diameter Penelitian 2, Jumlah Pasang Mating = 2

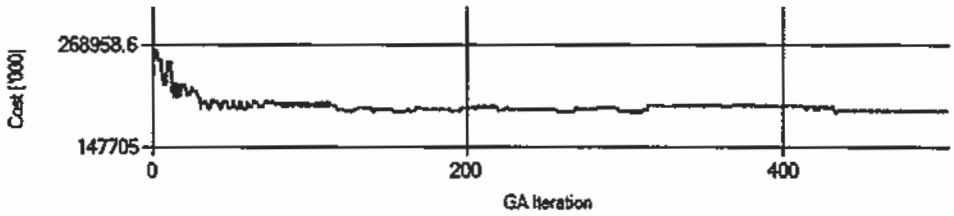
Path	Diameter (mm)	Harga/*path* ($)
1	200	17824800
2	150	16523200
3	200	2970800
4	200	17824800
5	200	3607400
6	100	17024800
7	250	35768400
8	300	10407600
9	250	14249200
10	200	33315400
11	300	18522000
12	150	15445600
	Total	203484000

Node	*Pressure* (bar)
A	
B	16,69
C	8,64
D	8,11
E	4,83
F	16,55
G	16,27
H	15,9
I	15,63
J	15,63
K	13,89
L	15,55
M	7,15
Min	4,83

- Jumlah Pasang Mating = 4

  Solusi:

  - Biaya = 192.268.080
  - Minimum Pressure = 2,60 bar
  - Grafik iterasi GA dalam mencari solusi ditunjukkan pada Gambar 4.16.



Gambar 4.16 Penelitian 2, Jumlah Pasang Mating = 4

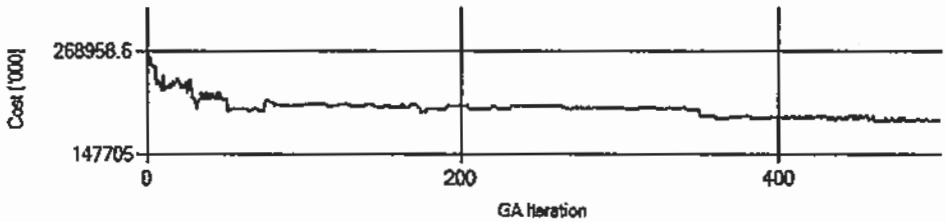
- Diameter yang digunakan pada tiap path ditunjukkan pada Tabel 4.19

Tabel 4.19 Hasil Diameter Penelitian 2, Jumlah Pasang Mating = 4

Path	Diameter (mm)	Harga/*path* ($)
1	200	17824800
2	200	19522400
3	200	2970800
4	200	17824800
5	200	3607400
6	100	17024800
7	250	35768400
8	200	7511880
9	200	10397800
10	150	28197200
11	200	13369600
12	200	18249200
	Total	192269080

Node	*Pressure* (bar)
A	
B	16,69
C	15,1
D	14,8
E	13,29
F	16,55
G	16,27
H	15,9
I	15,17
J	14,63
K	2,6
L	14,02
M	12,22
Min	2,6

- Jumlah Pasang Mating = 6
  Solusi:
  - Biaya = 188.195.080
  - Minimum Pressure = 4,63 bar
  - Grafik iterasi GA dalam mencari solusi ditunjukkan pada Gambar 4.17.



Gambar 4.17 Penelitian 2, Jumlah Pasang Mating = 6

- Diameter yang digunakan pada tiap path ditunjukkan pada Tabel 4.20.

Tabel 4.20 Hasil Diameter Penelitian 2, Jumlah Pasang Mating = 6

Path	Diameter (mm)	Harga/*path* ($)
1	200	17824800
2	150	16523200
3	200	2970800
4	200	17824800
5	250	4943600
6	100	17024800
7	300	36162000
8	200	7511880
9	200	10397800
10	150	28197200
11	200	13368600
12	150	15445600
	Total	188195080

Node	*Pressure* (bar)
A	
B	16,69
C	8,64
D	8,11
E	4,83
F	16,64
G	16,36
H	16,38
I	15,67
J	15,15
K	4,72
L	14,56
M	4,63
Min	4,63

Kesimpulan dari perubahan jumlah mating adalah :

Dari hasil penelitian, dapat ditarik kesimpulan bahwa semakin banyak pasangan yg mating maka semakin cepat algoritma genetik itu konvergen. Hal ini akan mengurangi kemungkinan kombinasi yang ada namun semakin cepat menemukan kondisi yang stabil. Konvergen diperlihat dari iterasi akhir yang selalu memberikan solusi yang sama. Dari penelitian jumlah mating mempengaruhi hasil akhir optimasi di nilai *cost.* Dengan banyaknya pasangan mating cost yang dihasilkan semakin optimal. Hasil pressure minimum yang dihasil node di jaringan juga berbeda namun tidak berhubungan langsung dengan jumlah mating.

4.4.3 Pengaruh Jumlah Mutasi dalam Kontribusinya terhadap Solusi

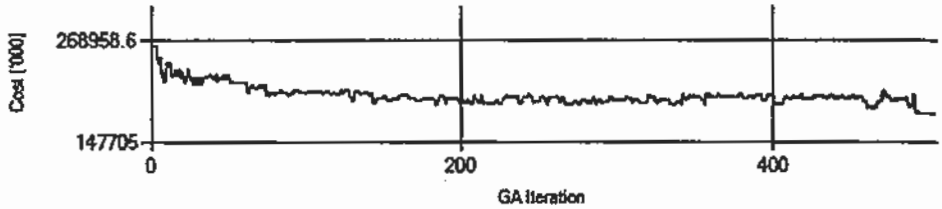
Parameter kunci yang digunakan yaitu:

- Jumlah Populasi: 30
- Batas Iterasi: 500
- Jumlah Mating: 2

Hasil penelitian:

- Jumlah Mutasi = 20%
  Solusi:

- Biaya = 182.665.480
- Minimum Pressure = 4,83 bar
- Grafik iterasi GA dalam mencari solusi ditunjukkan pada Gambar 4.18.



Gambar 4.18 Penelitian 3, Jumlah Mutasi = 20%

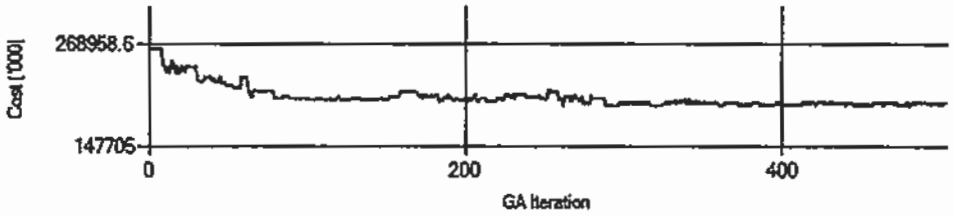
- Diameter yang digunakan pada tiap path ditunjukkan pada Tabel 4.21

Tabel 4.21 Hasil Diameter Penelitian 3, Jumlah Mutasi = 20%

Path	Diameter (mm)	Harga/*path* ($)
1	200	17824800
2	150	16523200
3	200	2970800
4	200	17824800
5	200	3607400
6	100	17024800
7	200	26100600
8	200	7511880
9	200	10397800
10	200	33315400
11	150	11314800
12	200	18249200
	Total	182665480

Node	*Pressure* (bar)
A	
B	16,69
C	8,64
D	8,11
E	4,83
F	16,55
G	16,27
H	14,55
I	13,75
J	13,15
K	11,03
L	10,09
M	7,39
Min	4,83

- Jumlah Mutasi = 40%
  Solusi:
  - Biaya = 200.681.280
  - Minimum Pressure = 8,73 bar
  - Grafik iterasi GA dalam mencari solusi ditunjukkan pada Gambar 4.18.



Gambar 4.19 Penelitian 3, Jumlah Mutasi = 40%

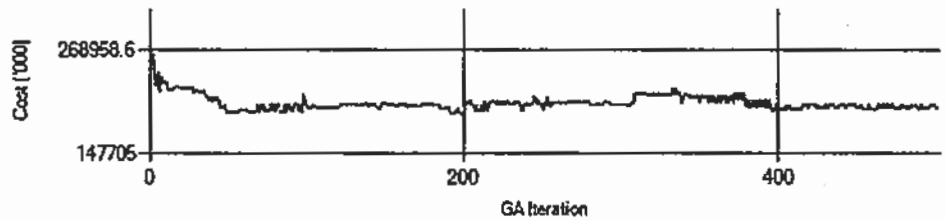
- Diameter yang digunakan pada tiap path ditunjukkan pada Tabel 4.22.

Tabel 4.22 Hasil Diameter Penelitian 3, Jumlah Mutasi = 40%

Path	Diameter (mm)	Harga/*path* ($)
1	200	17824800
2	200	19522400
3	300	4116000
4	200	17824800
5	250	4943600
6	200	22068800
7	200	26100600
8	200	7511880
9	150	8800400
10	200	33315400
11	200	13368600
12	300	25284000
	Total	200681280

Node	*Pressure* (bar)
A	
B	16,69
C	15,1
D	15,06
E	13,58
F	16,64
G	16,63
H	14,66
I	13,86
J	11,29
K	8,73
L	10,48
M	10,16
Min	8,73

- Jumlah Mutasi = 60%
  Solusi:
  - Biaya = 202.328.240
  - Minimum Pressure = 6,89 bar
  - Grafik iterasi GA dalam mencari solusi ditunjukkan pada Gambar 4.20.



Gambar 4.20 Penelitian 3, Jumlah Mutasi = 60%

- Diameter yang digunakan pada tiap path ditunjukkan pada Tabel 4.23

Tabel 4.23 Hasil Diameter Penelitian 3, Jumlah Mutasi = 60%

Path	Diameter (mm)	Harga/*path* ($)
1	250	24427200
2	150	16523200
3	250	4071200
4	200	17824800
5	250	4943600
6	150	18678400
7	250	35768400
8	150	6357840
9	150	8800400
10	200	33315400
11	200	13368600
12	200	18249200
	Total	202328240

Node	*Pressure* (bar)
A	
B	17,23
C	9,64
D	9,48
E	6,89
F	17,18
G	17,14
H	16,56
I	13,51
J	10,85
K	8,15
L	10,01
M	7,28
Min	6,89

Kesimpulan dari perubahan jumlah mating adalah :

Pada penelitian ini semakin banyak jumlah mutasi maka akan memperbanyak jumlah kombinasi. Sehingga untuk mencapai konvergensi yang cepat akan berkurang. Dari penelitian yang dilakukan ini tidak dapat terlihat biaya yang makin menurun akibat jumlah mutasi yang bertambah, dikarenakan banyaknya kombinasi baru yang keluar akibat mutasi dalam tiap iterasinya, sehingga menghasilkan harga minimum *cost* lokal di daerah lain. *Pressure* minimum yang terdapat pada node dalam skema jaringan juga tidak dipengaruhi oleh jumlah mutasi.

BAB 5

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Penelitian yang telah dilakukan menghasilkan kesimpulan sebagai berikut :

1. Hasil validasi perhitungan manual dan program yang sama untuk kasus skema jaringan sederhana dapat disimpulkan bahwa program berjalan sesuai dengan baik dan hasil program sesuai dengan harapan. Program dapat di gunakan untuk skema jaringan dengan batasan-batasan masalah yang telah ditentukan baik skema sederahana maupun kompleks.
2. Hasil penelitian dengan perubahan parameter- parameter dari program memperlihatkan hasil yang berbeda- beda. Kesimpulan yang dapat ditarik adalah :
    - Perubahan parameter jumlah populasi menghasilkan kesimpulan semakin besar jumlah populasi maka semakin besar kemungkinan Algoritma Genetik menghasilkan hasil yang optimum
    - Perubahan jumlah mating menghasilkan kesimpulan semakin banyak pasangan mating maka *cost* yang dihasilkan semakin minimal dan hasil lebih cepat konvergen.
    - Perubahan nilai mutasi menghasilkan kondisi penemuan *local minimum* yang lain sehingga banyaknya mutasi belum tentu menyebabkan *cost* yang dihasilkan minimal
3. Metode Algoritma Genetik ini dapat diterapkan untuk suatu model masalah penentuan penggunaan pipa dengan diamater yang tepat dalam jaringan perpipaan untuk setiap *path* dengan tetap dalam kondisi sesuai dengan batasan-batasan aturan dalam perpipaan. Dengan program yang dibangun dapat memperhitungkan biaya penggunaan pipa dengan perhitungan harga pipa untuk setiap diameter yang menjadi inputan untuk *user*.

5.2 Saran

Algoritma Genetik merupakan suatu metoda optimasi yang dapat diterapkan dalam berbagai bidang yang dinilai dapat dioptimalkan. Kunci dari penggunaan Algoritma Genetik adalah perumusan fungsi optimal untuk suatu kasus. Kesalahan yang fatal terjadi dalam optimasi dengan Algoritma Genetik apabila perumusan fungsi yang dibuat sebagai fungsi fitness salah.

Kemungkinan untuk pengembangan lebih lanjut penerapan Algoritma Genetik pada sistem jaringan perpipaan ini, dapat dibuat seaktual mungkin dengan memperhitungan peralatan lain dalam perpipaan dan dengan perhitungan yang lebih kompleks dengan melibatkan variable-variable lain. Sistem yang penulis bangun disini hanya merupakan simulasi sederhana yang menunjukan bahwa Algoritma Genetik dapat diterapkan dalam mencari solusi optimalisasi *cost* dari suatu jaringan dengan memperhitungkan penggunaan diameter pipa. Program dapat dikembangkan lebih lanjut dan bisa dijadikan acuan dari keputusan *real* dilapangan untuk masalah optimalisai perpipaan.

Pengembangan lebih lanjut juga dapat memperhitungkan hal – hal apa saja yang mungkin terjadi dalam suatu skema jaringan perpipaan. Misalnya di buat kemungkinan adanya loop, source yang lebih dari satu dan lainnya. Program selanjutnya diharapkan lebih memperhitungkan hal- hal lainnya.

Fleksibilitas dalam program yang penulis bangun sudah dibuat dengan fleksibilitas yang cukup, mungkin dimana berbagai perubahan yang dapat terjadi dapat dijadikan input. Program yang telah penulis bangun diharapkan dapat digunakan oleh siapa pun yang memerlukan dan dapat dikembangkan sedemikian rupa sesuai dengan kondisi real dilapangan.

DAFTAR PUSTAKA

1. Bhave, M.P. dan S.A. Patekar. *Programming with Java*. Dorling Kindersley Pvt. Ltd. India. 2009.
2. El-Mahdy, Omar F. Mohamed, Mohamed Ezz Hassan Ahmed, dan Sayed Metwalli. *Computer Aided Optimization of Natural Gas Pipe Networks using Genetic Algorithm*. Applied Soft Computing 10 (2010) 1141-1150, Elsevier B.V. 2008.
3. Fletcher, Roger. *Practical Methods of Optimization, 2nd Edition*. Great Britain: John Wiley & Sons, Inc. 2000.
4. Greanier, Todd. *Java Foundations*. Sybex Inc. 2004.
5. Haupt, Randy L. dan Sue Ellen Haupt. *Practical Genetic Algorithms, 2nd Edition*. New Jersey: John Wiley & Sons, Inc. 2004.
6. Liu, Henry. *Pipeline Engineering*. CRC Press LLC. 2003.
7. Mitchell, Melanie. *An Introduction to Genetic Algorithms*. London: MIT Press. 1999.
8. Rao, Singiresu S. *Engineering Optimization Theory and Practice*. New Jersey: John Wiley & Sons, Inc.2009

Lampiran 1

Hitungan Manual

Berikut ini merupakan hitungan manual yang digunakan untuk membandingkan kebeneran dari Algoritma Genetik.

inlet-A diameter	P2_A	cek pmin	A-B diameter	P2_B^2	cek P2_B > 0	P2_B	cek pmin	B-C diameter	P2_C^2	cek P2_C > 0	P2_C	cek Pmin	PRICE inlet-A diameter yang dipakai	PRICE A-B diameter yang dipakai	PRICE B-C diameter yang dipakai	cost inlet-A	cost A-B	cost B-A	total cost
100	84.98885	TRUE	100	52172474	TRUE	7223.052	TRUE	100	2.72197E+15	TRUE	52172474	TRUE	1650	1650	1650	28875000	13200000	20790000	62865000
100	84.98885	TRUE	100	52172474	TRUE	7223.052	TRUE	150	2.72197E+15	TRUE	52172474	TRUE	1650	1650	1790	28875000	13200000	22554000	64629000
100	84.98885	TRUE	100	52172474	TRUE	7223.052	TRUE	200	2.72197E+15	TRUE	52172474	TRUE	1650	1650	2950	28875000	13200000	37170000	79245000
100	84.98885	TRUE	100	52172474	TRUE	7223.052	TRUE	250	2.72197E+15	TRUE	52172474	TRUE	1650	1650	4000	28875000	13200000	50400000	92475000
100	84.98885	TRUE	150	52173125	TRUE	7223.097	TRUE	100	2.72203E+15	TRUE	52173125	TRUE	1650	1790	1650	28875000	14320000	20790000	63985000
100	84.98885	TRUE	150	52173125	TRUE	7223.097	TRUE	150	2.72203E+15	TRUE	52173125	TRUE	1650	1790	1790	28875000	14320000	22554000	65749000
100	84.98885	TRUE	150	52173125	TRUE	7223.097	TRUE	200	2.72203E+15	TRUE	52173125	TRUE	1650	1790	2950	28875000	14320000	37170000	80365000
100	84.98885	TRUE	150	52173125	TRUE	7223.097	TRUE	250	2.72203E+15	TRUE	52173125	TRUE	1650	1790	4000	28875000	14320000	50400000	93595000
100	84.98885	TRUE	200	52173204	TRUE	7223.102	TRUE	100	2.72204E+15	TRUE	52173204	TRUE	1650	2950	1650	28875000	23600000	20790000	73265000
100	84.98885	TRUE	200	52173204	TRUE	7223.102	TRUE	150	2.72204E+15	TRUE	52173204	TRUE	1650	2950	1790	28875000	23600000	22554000	75029000
100	84.98885	TRUE	200	52173204	TRUE	7223.102	TRUE	200	2.72204E+15	TRUE	52173204	TRUE	1650	2950	2950	28875000	23600000	37170000	89645000
100	84.98885	TRUE	200	52173204	TRUE	7223.102	TRUE	250	2.72204E+15	TRUE	52173204	TRUE	1650	2950	4000	28875000	23600000	50400000	102875000
100	84.98885	TRUE	250	52173221	TRUE	7223.103	TRUE	100	2.72205E+15	TRUE	52173221	TRUE	1650	4000	1650	28875000	32000000	20790000	81665000
100	84.98885	TRUE	250	52173221	TRUE	7223.103	TRUE	150	2.72205E+15	TRUE	52173221	TRUE	1650	4000	1790	28875000	32000000	22554000	83429000
100	84.98885	TRUE	250	52173221	TRUE	7223.103	TRUE	200	2.72205E+15	TRUE	52173221	TRUE	1650	4000	2950	28875000	32000000	37170000	98045000

100	84.98885	TRUE	250	52173221	TRUE	7223.103	TRUE	250	2.72205E+15	TRUE	52173221	TRUE	1650	4000	4000	28875000	32000000	50400000	111275000
														4000	4000	FALSE	FALSE	FALSE	FALSE
150	98.04089	TRUE	100	92390115	TRUE	9611.978	TRUE	100	8.53593E+15	TRUE	92390115	TRUE	1790	1650	1650	31325000	13200000	20790000	65315000
150	98.04089	TRUE	100	92390115	TRUE	9611.978	TRUE	150	8.53593E+15	TRUE	92390115	TRUE	1790	1650	1790	31325000	13200000	22554000	67079000
150	98.04089	TRUE	100	92390115	TRUE	9611.978	TRUE	200	8.53593E+15	TRUE	92390115	TRUE	1790	1650	2950	31325000	13200000	37170000	81695000
150	98.04089	TRUE	100	92390115	TRUE	9611.978	TRUE	250	8.53593E+15	TRUE	92390115	TRUE	1790	1650	4000	31325000	13200000	50400000	94925000
150	98.04089	TRUE	150	92390766	TRUE	9612.012	TRUE	100	8.53605E+15	TRUE	92390766	TRUE	1790	1790	1650	31325000	14320000	20790000	66435000
150	98.04089	TRUE	150	92390766	TRUE	9612.012	TRUE	150	8.53605E+15	TRUE	92390766	TRUE	1790	1790	1790	31325000	14320000	22554000	68199000
150	98.04089	TRUE	150	92390766	TRUE	9612.012	TRUE	200	8.53605E+15	TRUE	92390766	TRUE	1790	1790	2950	31325000	14320000	37170000	82815000
150	98.04089	TRUE	150	92390766	TRUE	9612.012	TRUE	250	8.53605E+15	TRUE	92390766	TRUE	1790	1790	4000	31325000	14320000	50400000	96045000
150	98.04089	TRUE	200	92390845	TRUE	9612.016	TRUE	100	8.53607E+15	TRUE	92390845	TRUE	1790	2950	1650	31325000	23600000	20790000	75715000
150	98.04089	TRUE	200	92390845	TRUE	9612.016	TRUE	150	8.53607E+15	TRUE	92390845	TRUE	1790	2950	1790	31325000	23600000	22554000	77479000
150	98.04089	TRUE	200	92390845	TRUE	9612.016	TRUE	200	8.53607E+15	TRUE	92390845	TRUE	1790	2950	2950	31325000	23600000	37170000	92095000
150	98.04089	TRUE	200	92390845	TRUE	9612.016	TRUE	250	8.53607E+15	TRUE	92390845	TRUE	1790	2950	4000	31325000	23600000	50400000	105325000
150	98.04089	TRUE	250	92390863	TRUE	9612.017	TRUE	100	8.53607E+15	TRUE	92390863	TRUE	1790	4000	1650	31325000	32000000	20790000	84115000
150	98.04089	TRUE	250	92390863	TRUE	9612.017	TRUE	150	8.53607E+15	TRUE	92390863	TRUE	1790	4000	1790	31325000	32000000	22554000	85879000
150	98.04089	TRUE	250	92390863	TRUE	9612.017	TRUE	200	8.53607E+15	TRUE	92390863	TRUE	1790	4000	2950	31325000	32000000	37170000	100495000
150	98.04089	TRUE	250	92390863	TRUE	9612.017	TRUE	250	8.53607E+15	TRUE	92390863	TRUE	1790	4000	4000	31325000	32000000	50400000	113725000
														4000	4000	FALSE	FALSE	FALSE	FALSE
200	99.51874	TRUE	100	98088070	TRUE	9903.942	TRUE	100	9.62127E+15	TRUE	98088070	TRUE	2950	1650	1650	51625000	13200000	20790000	85615000
200	99.51874	TRUE	100	98088070	TRUE	9903.942	TRUE	150	9.62127E+15	TRUE	98088070	TRUE	2950	1650	1790	51625000	13200000	22554000	87379000
200	99.51874	TRUE	100	98088070	TRUE	9903.942	TRUE	200	9.62127E+15	TRUE	98088070	TRUE	2950	1650	2950	51625000	13200000	37170000	101995000
200	99.51874	TRUE	100	98088070	TRUE	9903.942	TRUE	250	9.62127E+15	TRUE	98088070	TRUE	2950	1650	4000	51625000	13200000	50400000	115225000
200	99.51874	TRUE	150	98088721	TRUE	9903.975	TRUE	100	9.6214E+15	TRUE	98088721	TRUE	2950	1790	1650	51625000	14320000	20790000	86735000
200	99.51874	TRUE	150	98088721	TRUE	9903.975	TRUE	150	9.6214E+15	TRUE	98088721	TRUE	2950	1790	1790	51625000	14320000	22554000	88499000
200	99.51874	TRUE	150	98088721	TRUE	9903.975	TRUE	200	9.6214E+15	TRUE	98088721	TRUE	2950	1790	2950	51625000	14320000	37170000	103115000

200	99.51874	TRUE	150	98088721	TRUE	9903.975	TRUE	250	9.6214E+15	TRUE	98088721	TRUE	2950	1790	4000	51625000	14320000	50400000	116345000
200	99.51874	TRUE	200	98088801	TRUE	9903.979	TRUE	100	9.62141E+15	TRUE	98088801	TRUE	2950	2950	1650	51625000	23600000	20790000	96015000
200	99.51874	TRUE	200	98088801	TRUE	9903.979	TRUE	150	9.62141E+15	TRUE	98088801	TRUE	2950	2950	1790	51625000	23600000	22554000	97779000
200	99.51874	TRUE	200	98088801	TRUE	9903.979	TRUE	200	9.62141E+15	TRUE	98088801	TRUE	2950	2950	2950	51625000	23600000	37170000	112395000
200	99.51874	TRUE	200	98088801	TRUE	9903.979	TRUE	250	9.62141E+15	TRUE	98088801	TRUE	2950	2950	4000	51625000	23600000	50400000	125625000
200	99.51874	TRUE	250	98088818	TRUE	9903.98	TRUE	100	9.62142E+15	TRUE	98088818	TRUE	2950	4000	1650	51625000	32000000	20790000	104415000
200	99.51874	TRUE	250	98088818	TRUE	9903.98	TRUE	150	9.62142E+15	TRUE	98088818	TRUE	2950	4000	1790	51625000	32000000	22554000	106179000
200	99.51874	TRUE	250	98088818	TRUE	9903.98	TRUE	200	9.62142E+15	TRUE	98088818	TRUE	2950	4000	2950	51625000	32000000	37170000	120795000
200	99.51874	TRUE	250	98088818	TRUE	9903.98	TRUE	250	9.62142E+15	TRUE	98088818	TRUE	2950	4000	4000	51625000	32000000	50400000	134025000
														4000	4000	FALSE	FALSE	FALSE	FALSE
250	99.83734	TRUE	100	99350187	TRUE	9967.456	TRUE	100	9.87046E+15	TRUE	99350187	TRUE	4000	1650	1650	70000000	13200000	20790000	103990000
250	99.83734	TRUE	100	99350187	TRUE	9967.456	TRUE	150	9.87046E+15	TRUE	99350187	TRUE	4000	1650	1790	70000000	13200000	22554000	105754000
250	99.83734	TRUE	100	99350187	TRUE	9967.456	TRUE	200	9.87046E+15	TRUE	99350187	TRUE	4000	1650	2950	70000000	13200000	37170000	120370000
250	99.83734	TRUE	100	99350187	TRUE	9967.456	TRUE	250	9.87046E+15	TRUE	99350187	TRUE	4000	1650	4000	70000000	13200000	50400000	133600000
250	99.83734	TRUE	150	99350838	TRUE	9967.489	TRUE	100	9.87059E+15	TRUE	99350838	TRUE	4000	1790	1650	70000000	14320000	20790000	105110000
250	99.83734	TRUE	150	99350838	TRUE	9967.489	TRUE	150	9.87059E+15	TRUE	99350838	TRUE	4000	1790	1790	70000000	14320000	22554000	106874000
250	99.83734	TRUE	150	99350838	TRUE	9967.489	TRUE	200	9.87059E+15	TRUE	99350838	TRUE	4000	1790	2950	70000000	14320000	37170000	121490000
250	99.83734	TRUE	150	99350838	TRUE	9967.489	TRUE	250	9.87059E+15	TRUE	99350838	TRUE	4000	1790	4000	70000000	14320000	50400000	134720000
250	99.83734	TRUE	200	99350917	TRUE	9967.493	TRUE	100	9.8706E+15	TRUE	99350917	TRUE	4000	2950	1650	70000000	23600000	20790000	114390000
250	99.83734	TRUE	200	99350917	TRUE	9967.493	TRUE	150	9.8706E+15	TRUE	99350917	TRUE	4000	2950	1790	70000000	23600000	22554000	116154000
250	99.83734	TRUE	200	99350917	TRUE	9967.493	TRUE	200	9.8706E+15	TRUE	99350917	TRUE	4000	2950	2950	70000000	23600000	37170000	130770000
250	99.83734	TRUE	200	99350917	TRUE	9967.493	TRUE	250	9.8706E+15	TRUE	99350917	TRUE	4000	2950	4000	70000000	23600000	50400000	144000000
250	99.83734	TRUE	250	99350935	TRUE	9967.494	TRUE	100	9.87061E+15	TRUE	99350935	TRUE	4000	4000	1650	70000000	32000000	20790000	122790000
250	99.83734	TRUE	250	99350935	TRUE	9967.494	TRUE	150	9.87061E+15	TRUE	99350935	TRUE	4000	4000	1790	70000000	32000000	22554000	124554000
250	99.83734	TRUE	250	99350935	TRUE	9967.494	TRUE	200	9.87061E+15	TRUE	99350935	TRUE	4000	4000	2950	70000000	32000000	37170000	139170000
250	99.83734	TRUE	250	99350935	TRUE	9967.494	TRUE	250	9.87061E+15	TRUE	99350935	TRUE	4000	4000	4000	70000000	32000000	50400000	152400000
																			62865000

Lampiran 2

Kode Program

Berikut ini merupakan kode dari program.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace GAPipeline
{
  public partial class Form1 : Form
  {
    public Form1()
    {
      InitializeComponent();
    }

    private void buttonEncode_Click(object sender, EventArgs e)
    {
      int numPipeTypes = listView3.Items.Count;

      // Encode binary
      listView4.Items.Clear();
      for (int i = 0; i < listView3.Items.Count; i++)
      {
        listView4.Items.Add(listView3.Items[i].Text);
        listView4.Items[i].SubItems.Add(Convert.ToString(i, 2));
      }

      // Sesuaikan dijit dari hasil encoding
      for (int j = 0; j < listView4.Items.Count; j++)
      {
        for (int i = listView4.Items[j].SubItems[1].Text.Length; i <
listView4.Items[listView4.Items.Count - 1].SubItems[1].Text.Length; i++)
        {
          listView4.Items[j].SubItems[1].Text = "0" + listView4.Items[j].SubItems[1].Text;
        }
      }
    }

    private void buttonRunGA_Click(object sender, EventArgs e)
    {
      buttonEncode_Click(sender, e);

      tabControl1.SelectedTab = tabPage3;
      chart1.Series[0].Points.Clear();
```

```
        this.Refresh();

        listView5.Clear();
        listView8.Clear();
        listView6.Items.Clear();
        listView7.Items.Clear();
        listView9.Items.Clear();

        // Set minimum dan maksimum dari grafik
        int MinimumCostDiameter = Convert.ToInt32(listView3.Items[0].SubItems[1].Text);
        int MaksimumCostDiameter = Convert.ToInt32(listView3.Items[0].SubItems[1].Text);
        for (int i = 1; i < listView3.Items.Count; i++)
        {
          if (Convert.ToInt32(listView3.Items[i].SubItems[1].Text) < MinimumCostDiameter)
MinimumCostDiameter = Convert.ToInt32(listView3.Items[i].SubItems[1].Text);
          if (Convert.ToInt32(listView3.Items[i].SubItems[1].Text) > MaksimumCostDiameter)
MaksimumCostDiameter = Convert.ToInt32(listView3.Items[i].SubItems[1].Text);
        }
        int MinimumTotalCost = 0;
        int MaksimumTotalCost = 0;
        for (int i = 0; i < listView2.Items.Count; i++)
        {
          MinimumTotalCost += (MinimumCostDiameter *
Convert.ToInt32(listView2.Items[i].SubItems[3].Text));
          MaksimumTotalCost += (MaksimumCostDiameter *
Convert.ToInt32(listView2.Items[i].SubItems[3].Text));
        }
        chart1.ChartAreas[0].Axes[1].Minimum = (MinimumTotalCost/1000) - 1000;
        chart1.ChartAreas[0].Axes[1].Maximum = (MaksimumTotalCost*2/1000) + 2000;
        this.Refresh();

        // Set Kolom Iterasi GA and Kolom Pressure
        listView5.Columns.Add("Iteration");
        listView8.Columns.Add("Iteration");
        for (int j = 0; j < numericUpDown1.Value; j++)
        {
          for (int i = 0; i < listView2.Items.Count; i++)
          {
            listView5.Columns.Add(Convert.ToString(j + 1) + ". Path " + listView2.Items[i].Text);
          }
          for (int i = 1; i < listView1.Items.Count; i++)
          {
            listView8.Columns.Add(Convert.ToString(j + 1) + ". Node " + listView1.Items[i].Text);
          }
        }

        // Acak Populasi Awal
        listView5.Items.Add("0");
        int seed = Convert.ToInt32(numericUpDown6.Value);
        Random myRandom = new Random(seed);
        for (int j = 0; j < numericUpDown1.Value; j++)
        {
          for (int i = 0; i < listView2.Items.Count; i++)
          {

listView5.Items[0].SubItems.Add(listView4.Items[myRandom.Next(listView4.Items.Count)].SubItems[1].
Text);
          }
        }

        // Sesuaikan dijit dari populasi
```

```
        for (int j = 0; j < listView5.Columns.Count-1; j++)
        {
          for (int i = listView5.Items[0].SubItems[j+1].Text.Length; i <
listView4.Items[listView4.Items.Count - 1].SubItems[1].Text.Length; i++)
          {
            listView5.Items[0].SubItems[j + 1].Text = "0" + listView5.Items[0].SubItems[j + 1].Text;
          }
        }

        // Iterasi GA
        String SourceNode = listView1.Items[0].SubItems[0].Text;
        for (int iterateGA = 0; iterateGA < numericUpDown2.Value; iterateGA++)
        {
          // Hitung Pressure
          // Batasan masalah pressure source hanya 1
          listView8.Items.Add(Convert.ToString(iterateGA));
          for (int j = 0; j < numericUpDown1.Value; j++)
          {
            double[] p = new double[listView1.Items.Count - 1];
            double p1 = 0, p2 = 0;
            int[] d = new int[listView2.Items.Count];
            int l = 0;
            int[] q = new int[listView1.Items.Count - 1];
            String p1node = "", p2node = "";
            // Inisialisasi awal p
            for (int i = 0; i < listView1.Items.Count - 1; i++)
            {
              p[i] = 0;
              q[i] = Convert.ToInt32(listView1.Items[i+1].SubItems[2].Text);
            }
            // Inisialisasi d
            for (int i = 0; i < listView2.Items.Count; i++)
            {
              for (int k = 0; k < listView4.Items.Count; k++)
              {
                if (listView5.Items[iterateGA].SubItems[1 + i + (j * listView2.Items.Count)].Text ==
listView4.Items[k].SubItems[1].Text)
                {
                  d[i] = Convert.ToInt32(listView4.Items[k].SubItems[0].Text);
                  break;
                }
              }
            }
            // Lakukan perhitungan sebanyak jumlah path
            for (int i = 0; i < listView2.Items.Count; i++)
            {
              p1node = listView2.Items[i].SubItems[1].Text;
              p2node = listView2.Items[i].SubItems[2].Text;
              // Cari nilai p1
              if (p1node == SourceNode) p1 = (double)numericUpDown4.Value;
              else
              {
                for (int k = 1; k < listView1.Items.Count; k++)
                {
                  if (listView1.Items[k].SubItems[0].Text == p1node)
                  {
                    p1 = p[k - 1];
                  }
                }
              }
              // Hitung p2
```

```
                l = Convert.ToInt32(listView2.Items[i].SubItems[3].Text);
                p2 = Math.Pow(p1, 2) - ((19.43 * l / (Math.Pow(d[i], 4.854) * Math.Pow(0.9, 2))) *
Math.Pow(q[i], 1.854));
                p2 = Math.Pow(p2, 0.5);
                if (Double.IsNaN(p2))
                {
                   p2 = 0;
                }
                // Masukin ke daftar p
                for (int k = 1; k < listView1.Items.Count; k++)
                {
                   if (listView1.Items[k].SubItems[0].Text == p2node)
                   {
                      p[k - 1] = p2;
                      break;
                   }
                }
             }
             // Catat p
             for (int i = 0; i < listView1.Items.Count - 1; i++)
             {
                listView8.Items[iterateGA].SubItems.Add(Convert.ToString(p[i]));
             }
          }


          // Hitung Cost
          listView6.Items.Clear();
          // Iterasi sebanyak jumlah populasi
          for (int k = 0; k < numericUpDown1.Value; k++)
          {
             int Cost = 0;
             // Iterasi jumlah path
             for (int j = 0; j < listView2.Items.Count; j++)
             {
                // Iterasi tipe diameter dan cari harganya
                for (int i = 0; i < listView4.Items.Count; i++)
                {
                   if (listView5.Items[iterateGA].SubItems[(j + 1) + (k * listView2.Items.Count)].Text ==
listView4.Items[i].SubItems[1].Text)
                   {
                      Cost = Cost + Convert.ToInt32(listView3.Items[i].SubItems[1].Text) *
Convert.ToInt32(listView2.Items[j].SubItems[3].Text);
                   }
                }
             }
             listView6.Items.Add(Convert.ToString(k + 1));
             listView6.Items[k].SubItems.Add(Convert.ToString(Cost));
             listView6.Items[k].SubItems.Add("0");
             listView6.Items[k].SubItems.Add("0");
             listView6.Items[k].SubItems.Add("0");
          }

          // Hitung Penalti
          int MinCost = Convert.ToInt32(listView3.Items[0].SubItems[1].Text);
          int MaxCost = Convert.ToInt32(listView3.Items[0].SubItems[1].Text);
          int TotalLength = 0;
          int Penalti = 0;
          // Iterasi sebanyak jumlah populasi cari cost Minimum dan Maksimum
          for (int k = 1; k < listView3.Items.Count; k++)
          {
```

```
            if (Convert.ToInt32(listView3.Items[k].SubItems[1].Text) < MinCost)
              MinCost = Convert.ToInt32(listView3.Items[k].SubItems[1].Text);
            if (Convert.ToInt32(listView3.Items[k].SubItems[1].Text) > MaxCost)
              MaxCost = Convert.ToInt32(listView3.Items[k].SubItems[1].Text);
          }
          // Iterasi sebanyak jumlah Path
          for (int k = 0; k < listView2.Items.Count; k++)
          {
            TotalLength += Convert.ToInt32(listView2.Items[k].SubItems[3].Text);
          }
          // Cek jika ada yang melanggar aturan maka diberikan penalti
          Penalti = (MaxCost - MinCost) * TotalLength;
          for (int k = 0; k < numericUpDown1.Value; k++)
          {
            int TotalPenalti = 0;
            double MinPressure = Convert.ToDouble(numericUpDown5.Value);
            for (int j = 0; j < listView1.Items.Count - 1; j++)
            {
              double ptmp = Convert.ToDouble(listView8.Items[iterateGA].SubItems[1 + (k *
(listView1.Items.Count - 1)) + j].Text);

              if (ptmp < MinPressure)
              {
                TotalPenalti += Penalti;
              }
            }
            listView6.Items[k].SubItems[2].Text = Convert.ToString(TotalPenalti);
            listView6.Items[k].SubItems[3].Text =
Convert.ToString(Convert.ToInt32(listView6.Items[k].SubItems[1].Text) + TotalPenalti);
          }


          // Rank kromosom
          int[] indexRank = new int[listView6.Items.Count];
          for (int rank = 1; rank <= listView6.Items.Count; rank++)
          {
            int SearchCost = 0;
            int SearchIndex = 0;
            double SearchFitness = 0;

            // Inisialisasi awal dengan mencari yang belum dirank
            int i = 0;
            while ( i < listView6.Items.Count )
            {
              if (listView6.Items[i].SubItems[4].Text == "0")
              {
                SearchCost = Convert.ToInt32(listView6.Items[i].SubItems[3].Text);
                SearchFitness = 1 / (double) SearchCost;
                SearchIndex = i;
                break;
              }
              i++;
            }
            // Bandingkan dengan mencari yang terbesar
            for (i = SearchIndex+1; i < listView6.Items.Count; i++)
            {
              // Bandingkan jika belum mempunyai rank
              if (listView6.Items[i].SubItems[4].Text == "0")
              {
                if ((1 / (double) Convert.ToInt32(listView6.Items[i].SubItems[3].Text)) > SearchFitness)
                {
```

```
                    SearchCost = Convert.ToInt32(listView6.Items[i].SubItems[3].Text);
                    SearchFitness = 1 / (double) SearchCost;
                    SearchIndex = i;
                  }
                }
              }
              listView6.Items[SearchIndex].SubItems[4].Text = Convert.ToString(rank);
              indexRank[rank - 1] = SearchIndex;
            }

            // Gambar grafik

chart1.Series[0].Points.AddY(Convert.ToInt32(listView6.Items[indexRank[0]].SubItems[1].Text) / 1000);
            this.Refresh();

            // Lakukan mating, crossover
            int limitRank = (2 * Convert.ToInt32(numericUpDown3.Value));
            String[,] Mating = new String[limitRank, listView2.Items.Count];
            String[,] Offspring = new String[limitRank, listView2.Items.Count];
            int DivideCut = myRandom.Next(listView2.Items.Count);
            if (DivideCut == 0) DivideCut = listView2.Items.Count;
            int Cut = listView2.Items.Count / DivideCut;
            for (int rank = 0; rank < limitRank; rank++)
            {
              for (int i = 0; i < listView2.Items.Count; i++)
              {
                Mating[rank, i] = listView5.Items[iterateGA].SubItems[(i + 1) + (indexRank[rank] *
listView2.Items.Count)].Text;
              }
            }
            for (int couple = 0; couple < Convert.ToInt32(numericUpDown3.Value); couple++)
            {
              for (int i = 0; i < Cut; i++)
              {
                Offspring[(couple * 2), i] = Mating[(couple * 2), i];
                Offspring[((couple * 2) + 1), i] = Mating[(couple * 2) + 1, i];
              }
              for (int i = Cut; i < listView2.Items.Count; i++)
              {
                Offspring[(couple * 2), i] = Mating[((couple * 2) + 1), i];
                Offspring[((couple * 2) + 1), i] = Mating[(couple * 2), i];
              }
            }


            // Mutasi Populasi
            listView5.Items.Add(Convert.ToString(iterateGA + 1));
            for (int i = 1; i < listView5.Columns.Count; i++)
            {
              listView5.Items[iterateGA + 1].SubItems.Add(listView5.Items[iterateGA].SubItems[i].Text);
            }
            int Persentase = Convert.ToInt32(numericUpDown7.Value);
            int Populasi = Convert.ToInt32(numericUpDown1.Value);
            int Mutated = Populasi * Persentase / 100;
            //int Mutated = Convert.ToInt32();
            for (int i = 1; i <= Mutated; i++)
            {
              listView5.Items[iterateGA +
1].SubItems[myRandom.Next(Convert.ToInt32(listView2.Items.Count)) + 1 +
(indexRank[listView6.Items.Count - i] * listView2.Items.Count)].Text =
listView4.Items[myRandom.Next(listView4.Items.Count)].SubItems[1].Text;
```

```
            }

            // Offspring join ke Populasi
            for (int rank = 0; rank < limitRank; rank++)
            {
              for (int i = 0; i < listView2.Items.Count; i++)
              {
                listView5.Items[iterateGA + 1].SubItems[i + 1 + (indexRank[rank] *
listView2.Items.Count)].Text = Offspring[rank, i];
              }
            }
          }

          // Hitung Pressure Solusi
          // Batasan masalah pressure source hanya 1
          listView8.Items.Add(Convert.ToString(listView8.Items.Count));
          for (int j = 0; j < numericUpDown1.Value; j++)
          {
            double[] p = new double[listView1.Items.Count - 1];
            double p1 = 0, p2 = 0;
            int[] d = new int[listView2.Items.Count];
            int l = 0;
            int[] q = new int[listView1.Items.Count];
            String p1node = "", p2node = "";
            // Inisialisasi awal p
            for (int i = 0; i < listView1.Items.Count - 1; i++)
            {
              p[i] = 0;
              q[i] = Convert.ToInt32(listView1.Items[i + 1].SubItems[2].Text);
            }
            // Inisialisasi d
            for (int i = 0; i < listView2.Items.Count; i++)
            {
              for (int k = 0; k < listView4.Items.Count; k++)
              {
                if (listView5.Items[listView5.Items.Count-1].SubItems[1 + i + (j *
listView2.Items.Count)].Text == listView4.Items[k].SubItems[1].Text)
                {
                  d[i] = Convert.ToInt32(listView4.Items[k].SubItems[0].Text);
                  break;
                }
              }
            }
            // Lakukan perhitungan sebanyak jumlah path
            for (int i = 0; i < listView2.Items.Count; i++)
            {
              p1node = listView2.Items[i].SubItems[1].Text;
              p2node = listView2.Items[i].SubItems[2].Text;
              // Cari nilai p1
              if (p1node == SourceNode) p1 = (double)numericUpDown4.Value;
              else
              {
                for (int k = 1; k < listView1.Items.Count; k++)
                {
                  if (listView1.Items[k].SubItems[0].Text == p1node)
                  {
                    p1 = p[k - 1];
                  }
                }
              }
              // Hitung p2
```

```
            l = Convert.ToInt32(listView2.Items[i].SubItems[3].Text);
            p2 = Math.Pow(p1, 2) - ((19.43 * l / (Math.Pow(d[i], 4.854) * Math.Pow(0.9, 2))) *
Math.Pow(q[i], 1.854));
            p2 = Math.Pow(p2, 0.5);
            if (Double.IsNaN(p2))
            {
               p2 = 0;
            }
            // Masukin ke daftar p
            for (int k = 1; k < listView1.Items.Count; k++)
            {
               if (listView1.Items[k].SubItems[0].Text == p2node)
               {
                  p[k - 1] = p2;
                  break;
               }
            }
         }
         // Catat p
         for (int i = 0; i < listView1.Items.Count - 1; i++)
         {
            listView8.Items[listView8.Items.Count-1].SubItems.Add(Convert.ToString(p[i]));
         }
      }

      // Hitung Cost Solusi
      listView6.Items.Clear();
      // Iterasi sebanyak jumlah populasi
      for (int k = 0; k < numericUpDown1.Value; k++)
      {
         int Cost = 0;
         // Iterasi jumlah path
         for (int j = 0; j < listView2.Items.Count; j++)
         {
            // Iterasi tipe diameter dan cari harganya
            for (int i = 0; i < listView4.Items.Count; i++)
            {
               if (listView5.Items[listView5.Items.Count - 1].SubItems[(j + 1) + (k *
listView2.Items.Count)].Text == listView4.Items[i].SubItems[1].Text)
               {
                  Cost = Cost + Convert.ToInt32(listView3.Items[i].SubItems[1].Text) *
Convert.ToInt32(listView2.Items[j].SubItems[3].Text);
               }
            }
         }
         listView6.Items.Add(Convert.ToString(k + 1));
         listView6.Items[k].SubItems.Add(Convert.ToString(Cost));
         listView6.Items[k].SubItems.Add("0");
         listView6.Items[k].SubItems.Add("0");
         listView6.Items[k].SubItems.Add("0");
      }

      // Hitung Penalti Solusi
      int SolutionMinCost = Convert.ToInt32(listView3.Items[0].SubItems[1].Text);
      int SolutionMaxCost = Convert.ToInt32(listView3.Items[0].SubItems[1].Text);
      int SolutionTotalLength = 0;
      int SolutionPenalti = 0;
      // Iterasi sebanyak jumlah populasi cari cost Minimum dan Maksimum
      for (int k = 1; k < listView3.Items.Count; k++)
      {
         if (Convert.ToInt32(listView3.Items[k].SubItems[1].Text) < SolutionMinCost)
```

```
            SolutionMinCost = Convert.ToInt32(listView3.Items[k].SubItems[1].Text);
          if (Convert.ToInt32(listView3.Items[k].SubItems[1].Text) > SolutionMaxCost)
            SolutionMaxCost = Convert.ToInt32(listView3.Items[k].SubItems[1].Text);
        }
        // Iterasi sebanyak jumlah Path
        for (int k = 0; k < listView2.Items.Count; k++)
        {
          SolutionTotalLength += Convert.ToInt32(listView2.Items[k].SubItems[3].Text);
        }
        // Cek jika ada yang melanggar aturan maka diberikan penalti
        SolutionPenalti = (SolutionMaxCost - SolutionMinCost) * SolutionTotalLength;
        for (int k = 0; k < numericUpDown1.Value; k++)
        {
          int TotalPenalti = 0;
          double MinPressure = Convert.ToDouble(numericUpDown5.Value);
          for (int j = 0; j < listView1.Items.Count - 1; j++)
          {
            double ptmp = Convert.ToDouble(listView8.Items[listView5.Items.Count - 1].SubItems[1 + (k
* (listView1.Items.Count - 1)) + j].Text);

            if (ptmp < MinPressure)
            {
              TotalPenalti += SolutionPenalti;
            }
          }
          listView6.Items[k].SubItems[2].Text = Convert.ToString(TotalPenalti);
          listView6.Items[k].SubItems[3].Text =
Convert.ToString(Convert.ToInt32(listView6.Items[k].SubItems[1].Text) + TotalPenalti);
        }

        // Rank kromosom solusi
        int[] indexRankSolution = new int[listView6.Items.Count];
        for (int rank = 1; rank <= listView6.Items.Count; rank++)
        {
          int SearchCost = 0;
          int SearchIndex = 0;
          double SearchFitness = 0;

          // Inisialisasi awal dengan mencari yang belum dirank
          int i = 0;
          while (i < listView6.Items.Count)
          {
            if (listView6.Items[i].SubItems[4].Text == "0")
            {
              SearchCost = Convert.ToInt32(listView6.Items[i].SubItems[3].Text);
              SearchFitness = 1 / (double)SearchCost;
              SearchIndex = i;
              break;
            }
            i++;
          }
          // Bandingkan dengan mencari yang terbesar
          for (i = SearchIndex + 1; i < listView6.Items.Count; i++)
          {
            // Bandingkan jika belum mempunyai rank
            if (listView6.Items[i].SubItems[4].Text == "0")
            {
              if ((1 / (double)Convert.ToInt32(listView6.Items[i].SubItems[3].Text)) > SearchFitness)
              {
                SearchCost = Convert.ToInt32(listView6.Items[i].SubItems[3].Text);
                SearchFitness = 1 / (double)SearchCost;
```

```
                    SearchIndex = i;
                }
            }
        }
        listView6.Items[SearchIndex].SubItems[4].Text = Convert.ToString(rank);
        indexRankSolution[rank - 1] = SearchIndex;
    }

    // Gambar grafik

chart1.Series[0].Points.AddY(Convert.ToInt32(listView6.Items[indexRankSolution[0]].SubItems[1].Text) /
1000);
    this.Refresh();

    // Tulis diameter solusi
    int TotalSolutionCost = 0;
    int SolutionCost = 0;
    for (int i=0; i < listView2.Items.Count; i++)
    {
        listView7.Items.Add(Convert.ToString(i + 1));
        for (int j = 0; j < listView4.Items.Count; j++)
        {
            if (listView5.Items[listView5.Items.Count - 1].SubItems[1 + (indexRankSolution[0] *
listView2.Items.Count) + i].Text == listView4.Items[j].SubItems[1].Text)
            {
                listView7.Items[i].SubItems.Add(listView4.Items[j].SubItems[0].Text);
                SolutionCost = Convert.ToInt32(listView3.Items[j].SubItems[1].Text) *
Convert.ToInt32(listView2.Items[i].SubItems[3].Text);
                TotalSolutionCost += SolutionCost;
                listView7.Items[i].SubItems.Add(Convert.ToString(SolutionCost));
                break;
            }
        }
        //listView7.Items[i].SubItems.Add();
    }
    textBox1.Text = Convert.ToString(TotalSolutionCost);

    // Tulis pressure solusi
    listView9.Items.Add(listView1.Items[0].Text);
    for (int i = 0; i < listView1.Items.Count - 1; i++)
    {
        listView9.Items.Add(listView1.Items[i+1].Text);
        listView9.Items[i + 1].SubItems.Add(listView8.Items[listView8.Items.Count - 1].SubItems[1 +
(indexRankSolution[0] * (listView1.Items.Count - 1)) + i].Text);
    }

}

private void listView5_SelectedIndexChanged(object sender, EventArgs e)
{
    for (int n = 0; n < listView5.Items.Count; n++)
    {
        if (listView5.Items[n].Selected)
        {
            listView6.Items.Clear();
            // Iterasi sebanyak jumlah populasi
            for (int k = 0; k < numericUpDown1.Value; k++)
            {
                int Cost = 0;
                // Iterasi jumlah path
                for (int j = 0; j < listView2.Items.Count; j++)
```

```
                {
                    // Iterasi tipe diameter dan cari harganya
                    for (int i = 0; i < listView4.Items.Count; i++)
                    {
                        if (listView5.Items[n].SubItems[(j + 1) + (k * listView2.Items.Count)].Text ==
listView4.Items[i].SubItems[1].Text)
                        {
                            Cost = Cost + Convert.ToInt32(listView3.Items[i].SubItems[1].Text) *
Convert.ToInt32(listView2.Items[j].SubItems[3].Text);
                        }
                    }
                }
                listView6.Items.Add(Convert.ToString(k + 1));
                listView6.Items[k].SubItems.Add(Convert.ToString(Cost));
                listView6.Items[k].SubItems.Add("0");
                listView6.Items[k].SubItems.Add("0");
                listView6.Items[k].SubItems.Add("0");
            }

            // Hitung Penalti
            int MinCost = Convert.ToInt32(listView3.Items[0].SubItems[1].Text);
            int MaxCost = Convert.ToInt32(listView3.Items[0].SubItems[1].Text);
            int TotalLength = 0;
            int Penalti = 0;
            // Iterasi sebanyak jumlah populasi cari cost Minimum dan Maksimum
            for (int k = 1; k < listView3.Items.Count; k++)
            {
                if (Convert.ToInt32(listView3.Items[k].SubItems[1].Text) < MinCost)
                    MinCost = Convert.ToInt32(listView3.Items[k].SubItems[1].Text);
                if (Convert.ToInt32(listView3.Items[k].SubItems[1].Text) > MaxCost)
                    MaxCost = Convert.ToInt32(listView3.Items[k].SubItems[1].Text);
            }
            // Iterasi sebanyak jumlah Path
            for (int k = 0; k < listView2.Items.Count; k++)
            {
                TotalLength += Convert.ToInt32(listView2.Items[k].SubItems[3].Text);
            }
            // Cek jika ada yang melanggar aturan maka diberikan penalti
            Penalti = (MaxCost - MinCost) * TotalLength;
            for (int k = 0; k < numericUpDown1.Value; k++)
            {
                int TotalPenalti = 0;
                double MinPressure = (double)numericUpDown5.Value;
                for (int j = 0; j < listView1.Items.Count - 1; j++)
                {
                    double ptmp = Convert.ToDouble(listView8.Items[n].SubItems[1 + (k *
(listView1.Items.Count - 1)) + j].Text);

                    if (ptmp < MinPressure)
                    {
                        TotalPenalti += Penalti;
                    }
                }
                listView6.Items[k].SubItems[2].Text = Convert.ToString(TotalPenalti);
                listView6.Items[k].SubItems[3].Text =
Convert.ToString(Convert.ToInt32(listView6.Items[k].SubItems[1].Text) + TotalPenalti);
            }

            // Rank kromosom
            int[] indexRank = new int[listView6.Items.Count];
            for (int rank = 1; rank <= listView6.Items.Count; rank++)
```

```
            {
                int SearchCost = 0;
                int SearchIndex = 0;
                double SearchFitness = 0;

                // Inisialisasi awal dengan mencari yang belum dirank
                int i = 0;
                while (i < listView6.Items.Count)
                {
                    if (listView6.Items[i].SubItems[4].Text == "0")
                    {
                        SearchCost = Convert.ToInt32(listView6.Items[i].SubItems[3].Text);
                        SearchFitness = 1 / (double)SearchCost;
                        SearchIndex = i;
                        break;
                    }
                    i++;
                }
                // Bandingkan dengan mencari yang terbesar
                for (i = SearchIndex + 1; i < listView6.Items.Count; i++)
                {
                    // Bandingkan jika belum mempunyai rank
                    if (listView6.Items[i].SubItems[4].Text == "0")
                    {
                        if ((1 / (double)Convert.ToInt32(listView6.Items[i].SubItems[3].Text)) >
SearchFitness)
                        {
                            SearchCost = Convert.ToInt32(listView6.Items[i].SubItems[3].Text);
                            SearchFitness = 1 / (double)SearchCost;
                            SearchIndex = i;
                        }
                    }
                }
                listView6.Items[SearchIndex].SubItems[4].Text = Convert.ToString(rank);
                indexRank[rank - 1] = SearchIndex;
            }
            break;
        }
    }
}

private void button1_Click(object sender, EventArgs e)
{
    openFileDialog1.Filter = "Node files|*.txt";
    openFileDialog1.Title = "Select node list file";

    if (openFileDialog1.ShowDialog() == DialogResult.OK)
    {
        String[] tmpStr = new String[3];
        listView1.Items.Clear();

        System.IO.StreamReader sr = new System.IO.StreamReader(openFileDialog1.FileName);
        int n = 0;
        while (!(sr.EndOfStream))
        {
            tmpStr = sr.ReadLine().Split('\t');
            listView1.Items.Add(tmpStr[0]);
            listView1.Items[n].SubItems.Add(tmpStr[1]);
            listView1.Items[n].SubItems.Add(tmpStr[2]);
            n++;
        }
```

```
            sr.Close();
        }
    }

    private void button2_Click(object sender, EventArgs e)
    {
        openFileDialog1.Filter = "Path files|*.txt";
        openFileDialog1.Title = "Select path list file";

        if (openFileDialog1.ShowDialog() == DialogResult.OK)
        {
            String[] tmpStr = new String[4];
            listView2.Items.Clear();

            System.IO.StreamReader sr = new System.IO.StreamReader(openFileDialog1.FileName);
            int n = 0;
            while (!(sr.EndOfStream))
            {
                tmpStr = sr.ReadLine().Split('\t');
                listView2.Items.Add(tmpStr[0]);
                listView2.Items[n].SubItems.Add(tmpStr[1]);
                listView2.Items[n].SubItems.Add(tmpStr[2]);
                listView2.Items[n].SubItems.Add(tmpStr[3]);
                n++;
            }
            sr.Close();
        }
    }

    private void button3_Click(object sender, EventArgs e)
    {
        openFileDialog1.Filter = "Pipe Type files|*.txt";
        openFileDialog1.Title = "Select pipe type list file";

        if (openFileDialog1.ShowDialog() == DialogResult.OK)
        {
            String[] tmpStr = new String[2];
            listView3.Items.Clear();

            System.IO.StreamReader sr = new System.IO.StreamReader(openFileDialog1.FileName);
            int n = 0;
            while (!(sr.EndOfStream))
            {
                tmpStr = sr.ReadLine().Split('\t');
                listView3.Items.Add(tmpStr[0]);
                listView3.Items[n].SubItems.Add(tmpStr[1]);
                n++;
            }
            sr.Close();
        }
    }

}
}
```