e-ISSN 2962-3626

JURNAL KOMPUTER INDONESIA

Implementation Of A Genetic Algorithm To Determine The Minimum Spanning Tree (MST) In An Undirected Graph

Implementasi Algoritma Genetika Untuk Menentukan MST (*Minimum Spanning Tree*) Pada Graf Tak Berarah

Dhika Alfatah [1)]

[1)]*Program Studi Administrasi Publik, Sekolah Tinggi Ilmu Administrasi Negara*

Email: [1)] dhikaalfatah8@gmail.com

How to Cite :

Alfatah, D. (2025). Implementation Of A Genetic Algorithm To Determine The Minimum Spanning Tree (MST) In An Undirected Graph. Jurnal Komputer Indonesia, 4(1).

ARTICLE HISTORY

Received [25 Mei 2025]

Revised [28 Juni 2025]

Accepted [30 Juni 2025]

KEYWORDS

Algoritma, Genetika, Graf, Minimum Spanning Tree.

ABSTRAK

Penelitian ini dilatarbelakangi oleh adanya permasalahan *minimum spanning tree* di dalam graf yang sulit untuk diselesaikan sehingga membutuhkan metode tertentu untuk menyelesaikannya, di samping itu terdapat juga algoritma genetika yang dapat dijadikan sebagai metode untuk menyelesaikan permasalahan tersebut. Penelitian ini bertujuan untuk mengimplementasikan algoritma genetika untuk menentukan minimum spanning tree pada graf tak berarah. Metode perancangan sistem yang digunakan adalah Waterfall, sedangkan metode analisis yang digunakan adalah *Data Flow Diagram*. Untuk pengolahan data, metode yang digunakan adalah algoritma genetika. Sedangkan hasil penelitian diketahui bahwa algoritma genetika dapat diimplementasikan ke dalam aplikasi untuk menentukan *minimum spanning tree*

ABSTRACT

This research is background by existence of problems of minimum spanning tree in graph which difficult to be finished so that require certain method to finishing it, despitefully there are also algorithm of genetic able to be made as method to finish the problems. The purpose of this research is applying algorithm of genetic to determine minimum spanning tree at undirected graph. Method scheme of system the used is Waterfall, while analysis method the used is Data Flow Diagram. For the data processing of, method the used is algorithm of genetic. While result of research known that algorithm of genetic earn implementation into application to determine minimum spanning tree.

PENDAHULUAN

Graf merupakan salah satu cabang matematika yang muncul ketika seorang matematikawan Leonhart memecahkan masalah jembatan Koningsberg. Seiring dengan perkembangan bidang teknologi komputer, teori graf seringkali digunakan, baik konsep maupun aplikasinya. Salah satu permasalahan yang sering dibahas dalam graf adalah *Minimum Spanning Tree* sering disingkat dengan MST.

Minimum spanning tree suatu graf dipelajari sebagai materi dalam mata kuliah matematika diskrit di beberapa program studi, antara lain program studi matematika, ilmu komputer, dan teknik

informatika. Tidak banyak mahasiswa yang dapat dengan mudah memahami permasalahan *minimum spanning tree*.

Sebelum memahami pengertian *minimum spanning tree,* harus dipahami dulu apa yang dimaksud dengan graf. Graf adalah sekumpulan titik yang dihubungkan oleh sekumpulan garis. Terdapat dua jenis graf, yaitu graf sederhana (*simple graph*) dan graf tak sederhana (*unsimple graph*). Graf sederhana adalah graf yang tidak mengandung gelang maupun sisi (garis) ganda. Sedangkan graf tak sederhana adalah graf yang mengandung sisi ganda atau gelang. Yang akan dibahas dalam penelitian ini adalah graf sederhana. Jika dilihat dari arah garis (sisi) yang menghubungkan titik-titik graf, maka graf dapat dibagi menjadi dua jenis yaitu graf berarah dan graf tak berarah. Graf berarah artinya garis yang menghubungkan titik-titiknya mempunyai arah tertentu, sedangkan graf tak berarah sebaliknya.

Selanjutnya untuk memahami *minimum spanning tree*, maka perlu dipahami apa itu *tree* dan *spanning tree*. *Tree* adalah graf tak berarah terhubung yang tidak mengandung sirkuit. Jika sebarang garis yang menghubungkan titik tersebut ditelusuri maka akan kembali titik awal maka kumpulan titik dan garis tersebut disebut dengan sirkuit. Tree dapat dibentuk langsung atau diperoleh dari sebuah graf. Namun jika sebuah *tree* diperoleh dari suatu graf maka *tree* tersebut dinamakan dengan *spanning tree*. *Spanning tree* yang diperoleh dapat lebih dari satu bahkan banyak *spanning tree*. Tergantung dengan jumlah titik dalam graf tersebut. Semakin banyak titik dalam graf maka semakin banyak *spanning tree* yang dapat dibentuk.

Jika setiap sisi (garis) graf memiliki bobot, maka setiap spanning tree yang diperoleh dari graf tersebut, memiliki total bobot sisi yang tidak sama. Artinya ada yang lebih kecil dan ada yang lebih besar. Spanning tree yang memiliki bobot yang paling kecil dinamakan *minimum spanning tree* atau sering disingkat MST.

Tidak sulit untuk menentukan mana *spanning tree* yang memiliki bobot paling kecil, jika hanya ada beberapa *spanning tree* saja. Permasalahannya akan menjadi sulit untuk *minimum spanning tre*e, jika ditentukan dari banyak *spanning tree*. Permasalahan akan menjadi semakin sulit jika menentukan *minimum spanning tree* langsung dari suatu graf tanpa mengetahui total bobot setiap *spanning tree*. Pertanyaan yang muncul adalah bagaimana menentukan *minimum spanning tree* langsung dari suatu graf tanpa perlu mengetahui total bobot semua *spanning tree* tersebut?

Untuk menjawab pertanyaan tersebut diperlukan suatu aplikasi atau perangkat lunak yang dapat membantu menyelesaikan permasalahan tersebut. Namun perangkat lunak tidak serta merta dapat dibuat dan langsung jadi jika diketahui teknik apa yang dapat menentukan *minimum spanning tree* (MST) tersebut. Ada beberapa teknik yang ditawarkan untuk menentukan *minimum spanning tree* (MST), di antaranya teknik dengan algoritma Prim dan algoritma Kruskal. Namun pada pelaksanaannya, algoritma Prim tidak selalu memberikan solusi yang optimal dalam menentukan *minimum spanning tree* dari suatu graf. Sedangkan algoritma Kruskal dapat memberikan solusi optimal namun membutuhkan waktu yang lama dalam prosesnya menentukan *minimum spanning tree*. Selain kedua algoritma tersebut pilihan lain adalah teknik dengan algoritma genetika yang diyakini dapat menentukan *minimum spanning tree* (MST) secara optimal, karena algoritma genetika mendapatkan solusi dari calon-calon solusi yang dipilih secara acak dari semua calon solusi yang ada pada *minimum spanning tree* tersebut.

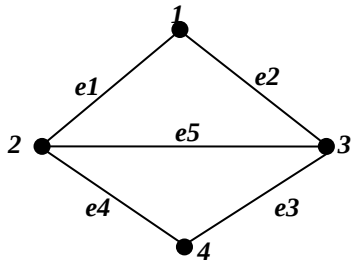
LANDASAN TEORI

Graf

Secara matematis, Graf *G* didefinisikan sebagai pasangan himpulan *(V,E),* ditulis dengan notasi *G=(V,E)*, yang dalam hal ini *V* adalah himpunan tidak kosong dari simpul-simpul (*vertices* atau *nodes*) dan *E* adalah himpulan sisi (*edges* atau *arcs*) yang menghubungkan sepasang simpul (Munir, 2007: 356).

e-ISSN 2962-3626
JURNAL KOMPUTER INDONESIA

Secara geometris graf dapat digambarkan sebagai sekumpulan noktah (simpul) di dalam bidang yang dihubungkan dengan sekumpulan garis (sisi).



Gambar 1 Contoh Graf

Gambar 1 memperlihatkan satu buah graf G dengan himpunan simpul V dan himpunan sisi E adalah:

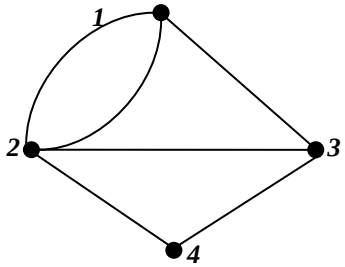
$V = \{1, 2, 3, 4\}$

$E = \{(1,2), (1,3), (2,3), (2,4), (3,4)\}$ atau

$E = \{e1, e2, e3, e4, e5\}$

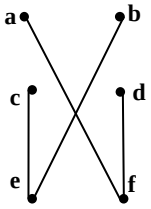
Di dalam graf dikenal istilah-istilah yang melekat dan menjadi sifat dari graf tersebut, yaitu:

1. Lintasan (*path*)
   Menurut Munir (2007: 369), Lintasan yang panjangnya *n* dari simpul awal v_0 di dalam graf G ialah barisan berselang-seling simpul dan sisi-sisi yang berbentuk $v_0, e_1, v_1, e_2, \ldots, v_{n-1}, e_n, v_n$ sedemikian sehingga e_1=(v_0, v_1), e_2=(v_1, v_2),..., e_n=(v_{n-1}, v_n) adalah sisi dari graf G.
2. Siklus (*Cycle*) atau sirkuit (*Circuit*)
   Menurut munir (2007: 370), Lintasan yang berawal dan berakhir pada simpul yang sama disebut sirkuit atau siklus. Gambar 2.2, menunjukkan graf yang memiliki sirkuit.



Gambar 2 Graf yang memiliki sirkuit

3. Terhubung (*connected*)
   Keterhubungan dua buah simpul adalah penting di dalam graf. Dua buah simpul *u* dan simpul *v* dikatakan terhubung jika terdapat lintasan dari *u* ke *v*. jika dua buah simpul terhubung maka pasti simpul yang pertama dapat dicapai dari simpul kedua. Gambar 2.1, dan 2.2 merupakan graf terhubung, sedangkan gambar 2.3 merupakan graf yang tak terubung.



Gambar 3 Graf tak terhubung

4. Graf berbobot (*weighted graf*)

   Menurut Munir (2007: 376), graf berbobot adalah graf yang setiap sisinya diberi sebuah harga (bobot). Bobot pada tiap sisi dapat berbeda-beda bergantung pada masalah yang dimodelkan

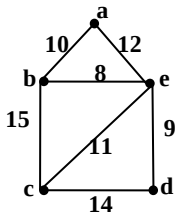
dengan graf. Bobot dapat menyatakan jarak antara dua buah kota, biaya perjalanan antara dua buah kota, waktu tempuh pesan (*message*) dari sebuah simpul komunikasi ke simpul komunikasi lain (dalam jaringan komputer), ongkos produksi, dan sebagainya.

Graf Tak berarah (*undirected graph*)

Menurut Munir (2007: 358), graf yang sisinya tidak mempunyai orientasi arah disebut graf tak berarah. Jadi, (*u,v*) =(*v,u*) adalah sisi yang sama. Pada graf tak berarah, urutan pasangann simpul yan dihubungkan tidak diperhatikan. Pada gambar 2.1 di atas adalah contoh graf tak berarah. Jika dilihat dari bobot sisi-sisinya, maka graf tak berarah juga dapat dibagi menjadi dua kategori, yaitu:

Graf tak berarah berbobot

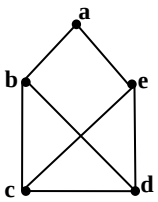
Graf tak berarah dikatakan berbobot jika setiap sisi-sisinya diberikan bobot atau nilai tertentu. Contoh, lihat gambar 4.



Gambar 4 Graf tak berarah berbobot

Graf tak berarah tak berbobot

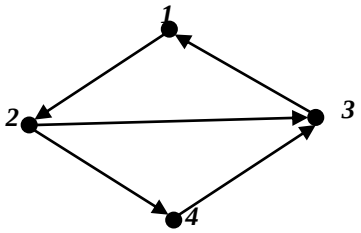
Graf tak berarah dikatakan tak berbobot, jika sisi-sisi pada graf tidak diberikan bobot atau nilai tertentu. Contoh graf tak berarah tak berbobot pada gambar 2.5.



Gambar 5 Graf tak berarah tak berbobot

Graf Berarah (*directed graph* atau *digraph*)

Menurut Munir (2007: 358), Graf yang setiap sisinya diberikan orientasi arah disebut graf berarah. Pada graf berarah, (*u,v*) dan (*v,u*) menyatakan dua buah sisi yang berbeda, dengan kata lain (*u,v*) ≠ (*v,u*). untuk sisi (*u,v*), *u* dikatakan simpul asal (*inisial vertex*) dan *v* simpul terminal (*terminal vertex*). Contoh graf berarah lihat gambar 2.6.



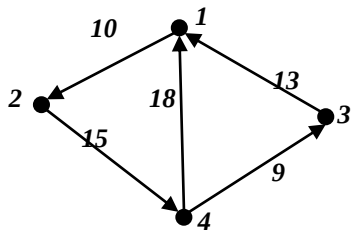
Gambar 6 Graf berarah

Seperti halnya graf tak berarah, graf berarah juga dapat dibagi menjadi dua kategori, yaitu:

Graf berarah berbobot

Graf berarah dikatakan berbobot jika pada setiap sisinya diberikan bobot atau nilai tertentu. Lihat gambar 7

e-ISSN 2962-3626
JURNAL KOMPUTER INDONESIA



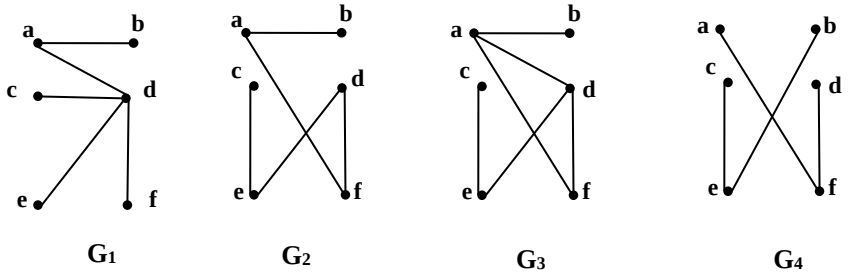
Gambar 7 Graf berarah berbobot

Graf berarah tak berbobot

Graf berarah dikatakan tak berbobot jika pada sisi-sisinya tidak diberikan bobot atau nilai tertentu. Contoh graf berarah tak berbobot dapat dilihat pada gambar 2.6 di atas.

Pohon (*Tree*)

Pohon (Tree) adalah graf tak-berarah terhubung tidak mengandung sirkuit (Munir, 2007: 444). Berdasarkan definisi tersebut ada dua sifat penting pada pohon, yaitu terhubung dan tidak mengandung sirkuit. Pada gambar 2.8 menunjukkan graf yang merupakan *tree* dan yang bukan *tree*.



Gambar 8 Tree dan Bukan Tree

Hanya G_1 dan G_2 yang merupakan *tree*, sedangkan G_3 dan G_4 bukan tree. G_3 bukan *tree* karena ia mengandung sirkuit *a, d , f, a,* sedangkan G4 bukan tree karena tidak terhubung, tetapi hanya bersilangan dua buah sisi, dalam hal ini sisi (*a,f*) dan sisi (*b,e*) dan tidak menyatakan suatu simpul. Pohon (*tree*) dapat dibentuk dari sebuah graf tak berarah dan terhubung dengan cara memutuskan satu atau lebih garis yang menghubungkan titik-titiknya. Pohon (*tree*) yang diperoleh dengan cara seperti ini disebut dengan pohon merentang (*spanning tree*).

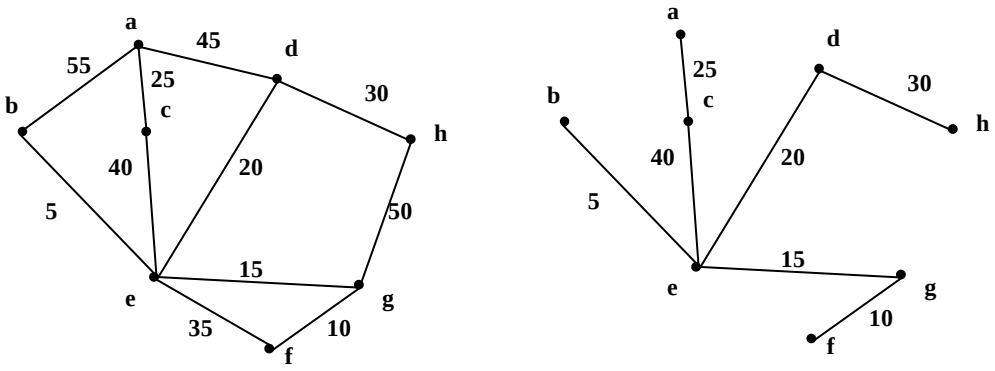
Pohon Merentang (*Spanning Tree*)

Misalkan *G=(V,E)* adalah graf tak-berarah terhubung yang bukan pohon, yang berati di G terdapat beberapa sirkuit. G dapat diubah menjadi pohon $T=(V_1,E_1)$ dengan cara memutuskan sirkuit-sirkuit yang ada (Munir, 2007: 447). Caranya mula-mula dipilih sebuah sirkuit, lalu hapus satu buah sisi dari sirkuit tersebut. G akan tetap terhubung dan jumlah sirkuit berkurang satu. Bila proses dilakukan berulang-ulang sampai semua sirkuit di G hilang, maka G menjadi sebuah pohon T, yang dinamakan pohon merentang (*spanning tree*) (Munir, 2007:447). Disebut pohon merentang karena semua simpul pada T sama dengan simpul pada graf G, dan sisi pada pohon $T \subseteq$ sisi pada graf G, dengan kata lain *V1=V* dan $E1 \subseteq E$. Jika setiap sisi masing-masing pohon merentang (*spanning tree*) tersebut memiliki bobot, maka total bobot masing-masing pohon merentang tidaklah sama. Dengan demikian tentu ada pohon merentang dengan bobot paling kecil dan dan ada pohon merentang dengan bobot yang paling besar. Pohon merentang (*spanning tree*) yang memiliki bobot terkecil dinamakan dengan pohon merentang minimum (*minimum spanning tree*).

Pohon Merentang Minimum (*Minimum Spanning Tree*)

Jika *G* adalah graf berbobot, maka bobot pohon merentang *T* dari *G* didefinisikan sebagai jumlah bobot semua sisi di *T*. Pohon merentang yang berbeda mempunyai bobot berbeda pula. Di antara semua pohon merentang di G, pohon merentang yang berbobot minimum dinamakan

pohon merentang minimum (*minimum spanning tree*). Minimum spanning tree (MST) merupakan *tree* yang paling penting, karena memiliki banyak terapan di kehidupan sehari-hari. Misalnya, pemerintah akan membangun jalur kereta rel api yang menghubungkan sejumlah kota seperti yang digambarkan oleh graf pada gambar 2.9.



Gambar 9 Graf dan Tree yang menyatakan jalur kereta api

Membagun jalur kereta api biayanya mahal, karena itu pembangunan jalur tersebut tidak perlu menghubungkan langsung dua buah kota; tetapi cukup membangun jalur kereta seperti pohon merentang. Karena itu dalam sebuah graf mungkin saja terdapat lebih dari satu pohon merentang (*spanning tree*), harus dicari pohon merentang yang mempunyai jumlah jarak terpendek, dengan kata lain harus dicari pohon merentang minimum.

Untuk menentukan *minimum spanning tree* dari suatu graf dibutuhkan suatu algoritma tertentu. Di dalam matematika diskrit ada dua teorema yang dipelajari yaitu algoritma prim dan algoritma kruskal. Algoritma dapat menentukan *minimum spanning tree* dari suatu graf namun terkadang tidak secara optimal. Sedangkan algoritma kruskal dapat menentukan minimun spanning tree namun memerlukan waktu yang cukup lama dalam prosesnya.

Selain kedua algoritma di atas, selanjutnya akan dijelaskan algoritma lain yang dapat dijadikan sebagai algoritma alternatif dalam menentukan *minimum spanning tree*. Algoritma yang dimaksud adalah algoritma genetika. Karena algoritma genetika dapat menyelesaikan berbagai masalah optimasi, maka algoritma genetika dapat digunakan juga untuk menentukan *minimum spanning tree* (MST).

Algoritma Genetika

Menurut Kusumadewi dan Purnomo (2005: 231), Algoritma genetika adalah algoritma pencarian heuristik yang didasarkan atas mekanisme evolusi biologis. Keberagaman pada evolusi biologis adalah variasi dari kromosom antar individu organisme. Variasi kromosom ini akan mempengaruhi laju reproduksi dan tingkat kemampuan organisme untuk tetap hidup. Pada dasarnya ada empat kondisi yang sangat mempengaruhi proses evolusi, yaitu:

a. Kemampuan organisme untuk melakukan reproduksi
b. Keberadaan populasi organisme yang bisa melakukan reproduksi
c. Keberagaman organisme suatu populasi
d. Perbedaan kemampuan untuk survive.

Individu yang lebih kuat (fit) akan memiliki tingkat survival dan tingkat reproduksi yang lebih tinggi jika dibandingkan dengan individu yang kurang fit. Pada kurun waktu tertentu (dikenal dengan istilah generasi), populasi secara keseluruhan akan lebih banyak memuat organisme yang kurang fit.

Komponen-Komponen Utama Algoritma Genetika

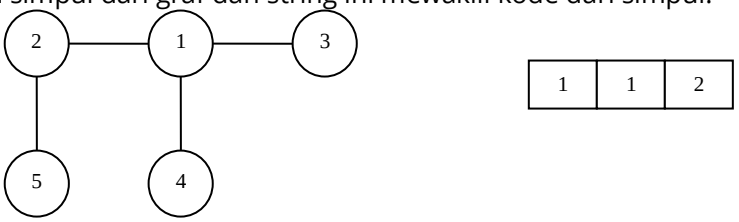
Ada enam komponen utama dalam algoritma genetika, yaitu teknik Penyandian, Prosedur Inisialisasi, Fungsi Evaluasi, Seleksi, Operator Genetika dan Penentuan Parameter (Kusumadewi dan Purnomo, 2005: 232).

Teknik Penyandian (Pengkodean)

Teknik Penyandian di sini meliputi penyandian gen dan kromosom. Gen merupakan bagian dari kromosom. Satu gen biasanya mewakili satu variabel. Gen dapat direpresentasikan dalam bentuk: string bit, pohon, array bilangan real, daftar aturan, elemen permutasi, elemen program, dan lain-lain. Contoh string bit: 10111, 01101, 11101, dst.

Model pengkodean calon solusi untuk setiap masalah berbeda-beda, adapun pengkodean untuk menentukan *minimum spanning tree* (MST) ini menggunakan pengkodean *prufer number*. Pengkodean prufer number ini bertujuan untuk mengkodekan pohon-pohon (*trees*) yang dihasilkan dari sebuah graf, agar masalah *minimum spanning tree* dapat diselesaikan dengan algoritma genetika. Adapun langkah-langkah pengkodean (*encoding*) prufer number dari sebuah pohon antara lain:

a) Misal simpul *i* merupakan simpul bernomor paling kecil dan ruas yang terhubung dengan simpul *i* tersebut hanya ada 1.
b) Pengkodean disusun dari kiri ke kanan. Misal *j* adalah digit pertama hasil pengkodean, di mana *j* merupakan simpul yang terhubung langsung ke *i,* maka digit selanjutnya diletakkan di sebelah kanan *j*
c) Buang simpul *i* dan ruas, dan ruas yang menghubungkan simpul I dan j, sehingga pohon tersebut hanya mengandung n-1 simpul.
d) Ulangi langkah di atas sampai ruas yang tertinggal hanya ada 1. Hasil akhir dari pengkodean ini disebut dengan prufer number dan berupa kumpulan string sebanyak n-2, di mana n adalah jumlah simpul dari graf dan string ini mewakili kode dari simpul.



Gambar 10. Proses Encoding

Gambar (a) merupakan representasi sebuah *tree* (pohon) dan (b) merupakan hasil *encoding* (penyandian).

Prosedur inisialisasi

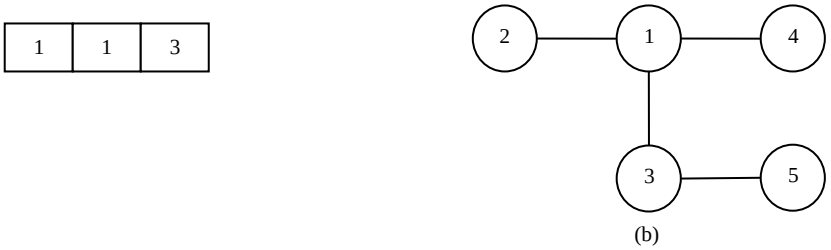
Inisialisasi kromosom dilakukan secara acak, namun harus tetap memperhatikan domain solusi dan kendala permasalahan yang ada. Berdasarkan teorema Cayley's bahwa sebuah graf dapat dihasilkan sebanyak $n^{(n-2)}$ pohon yang berbeda, maka apabila dalam menentukan seluruh pohon dilakukan satu persatu akan membutuhkan waktu yang cukup lama. Oleh karena itu, pembangkitan populasi awal dilakukan scara acak sebanyak n kromosom. Setiap kromosom merupakan hasil pengkodean dari pohon-pohon yan berbeda di mana nilai *fitness*-nya paling kecil.

Fungsi Evaluasi

Ada dua hal yang harus dilakukan dalam melalukan evaluasi kromosom, yaitu: evaluasi objektif (fungsi tujuan) dan konversi fungsi objektif ke dalam fungsi fitness.

Kromosom-kromosom yang dihasilkan baik dari pembangkita populasi awal atau crossover akan dievaluasi terlebih dahulu. Adapun proses yang terjadi saat evaluasi adalah *decoding* kromosom-kromosom dengan langkah-langkah pada decoding adalah:

1) Andaikan P adalam Prufer number danQ adalah himpulan simpul yang tidak terdapat pada P. Q disebut sebagai *eligible vertex*.
2) Misal *i* adalah satu simpul dari *eligible vertex* di mana *i* merupakan simpul dengan penomoran terkecil, dan *j* adalah digit paling kiri dariP. Tambahkan ruas daru *i* ke *j*, buang *i* dari Q dan j dari P, jika tidak ada lagi di P, letakkan *j* pada Q. Ulangi proses sampai tidak ada lagi digit yang tinggal di P.
3) Jika tidak ada digit di P, berarti hanya ada tepat 2 eligible vertex lagi yaitu *r* dan *s*. tambahkan ruas dari *r* ke *s* sehingga pada akhirnya membentuk sebuah pohon dengan *n-1* ruas.



Gambar 11. Proses Decoding

Gambar 11 merupakan gambaran proses *decoding, (a)* merupakan *prufer number* sedangkan (b) merupakan pohon hasil *decoding*. *Decoding* bertujuan mengubah setiap kromosom menjadi sebuah pohon sehingga jumlah bobot seluruh dari setiap pohon yang berbentuk dapat dihitung. Secara matematika, untuk menghitung jumlah bobot (*weight*) seluruh ruas dari setiap pohon diformulasikan

Visual Basic

Gambaran Umum Visual Basic

Menurut Swastika (2007: 2), Visual Basic (VB) adalah sebuah aplikasi pemrograman yang basis bahasa pemrogramannya adalah BASIC. BASIC merupakan bahasa pemrograman yang telah digunakan lebih dari 35 tahun, yang sering digunakan oleh pemula yang ingin belajar bahasa pemrograman, karena kemudahan dan kemiripan perintah dengan bahasa manusia. Bahasa pemrograman seperti pascal atau C, akan membutuhkan lebih banyak waktu untuk menguasainya, dibandingkan BASIC. Swastika (2007: 3) menambahkan, jika pemakai ingin membuat aplikasi windows dengan menggunakan VB, maka pemakai dapat melakukan langkah-langkah berikut:

1. Tentukan apa yang hendak dilakukan program dengan merancang keseluruhan desain.
2. Kerjakan bagian visual dari aplikasi/program tersebut (layar dan menu di mana user akan berinteraksi).
3. Tambahkan kode program VB, yang sesuai untuk masing-masing elemen visual yang ada untuk mengotomatisasi program.
4. Uji aplikasi/program untuk menemukan bug program. Bug adalah istilah untuk kesalahan program. Jika sebuah program yang ditulis tidak bekerja dengan apa yang diharapkan, maka harus dilakukan *debugging* untuk mengetahui kesalahan program dan memperbaikinya.
5. Kompilasi aplikasi/program yang telah dites tadi.

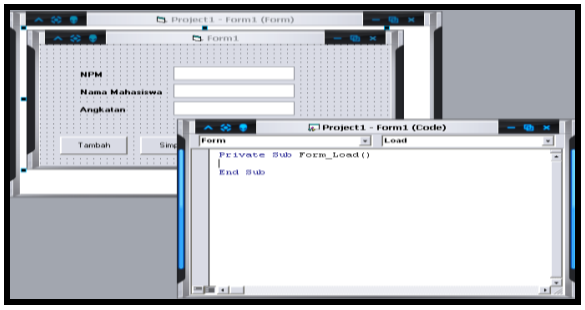
Komponen Visual Basic

Menurut Swastika (2007: 4), sebagian besar kepopuleran VB adalah karena Integrated Development Environment (atau IDE), yang sangat mempermudah pembuatan program. Teorinya, program VB tak lebih dari sekumpulan kode yang disusun dengan aturan tertentu.

e-ISSN 2962-3626
JURNAL KOMPUTER INDONESIA

Dengan mengunakan IDE, programer tidaknya hanya dipermudah dalam mengetikkan kode-kode program, tapi programer juga dapat langsung menguji (dan menjalankan) program, melakukan *debugging* untuk mencari kesalahan pada program dan mengompilasi program menjadi file executable, agar dapat dijalankan langsung tanpa melalui IDE.

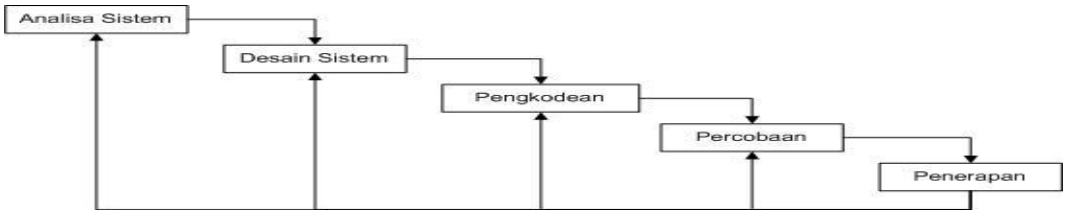
Form Designer window adalah tempat di mana programer membuat tampilan aplikasi. Seperti kanvas pada sebuah lukisan, form inilah yang nantinya dihadapi oleh pengguna dari dari aplikasi yang dibuat *programer*. Dalam sebuah aplikasi dapat terdiri dari banyak form. Masing-masing form mempunyai kode program yang diletakkan dalam *Code Window*. Pada *code windo*w inilah program diatur, seperti apa yang harus dijalankan oleh sebuah ketika sebuah button di-klik, apa yang harus dijalankan saat pengguna menekan Enter, dan sebagainya. Untuk menampilkan form designer, dapat dilkaukan dengan men-double klik pada *project window*. *Code window* dapat ditampilkan dengan men-double klik pada *Form designer* atau mengunakan shortcut F7.



Gambar 12. Form designer dan Code window

METODE PENELITIAN

Dalam penelitian skripsi skripsi ini penulis mengumpulkan dengan metode Studi Pustaka yaitu suatu metode untuk mendapatkan informasi yang berhubungan dengan penelitian ini, dengan cara membaca dan mempelajari buku-buku atau teori-teori yang berkaitan dengan penelitian yang dilakukan. Metode perancangan sistem yang digunakan dalam penelitian adalah metode Waterfall, yaitu sebuah metode yang pengerjaan suatu sistem dilakukan secara berurutan atau secara linear. Jadi jika langkah satu belum dikerjakan maka tidak akan dapat melakukan pengerjaan langkah 2, 3 dan seterusnya. Secara otomatis tahapan ke-3 akan bisa dilakukan jika tahap ke-1 dan ke-2 sudah dilakukan. Secara garis besar metode waterfall mempunyai langkah-langkah adalah: Analisa Sistem, Desain Sistem, Pengkodean, Percobaan, dan Penerapan.



Gambar 13 Model Waterfall

HASIL DAN PEMBAHASAN

Database

Aplikasi untuk Menentukan *Minimum Spanning Tree* (MST) pada Graf Tak Berarah ini melibatkan suatu *database* dengan dua tabel, yaitu Simpul dan JarakSimpul, oleh karena itu proses pembuatan database tersebut perlu dijelaskan. *Database* yang digunakan diletakkan pada MySql Server 5.1 yang terintegrasi ke dalam perangkat lunak Xampp. Pembuatan *database* sendiri

dilakukan dengan bantuan PHPMyAdmin. Untuk mengaktifkan PhpMyAdmin dilakukan dengan mengaktifkan http://localhost/phpmyadmin, selanjutnya menekan tab Basisdata diteruskan dengan menuliskan nama database DataNeviMST, seperti tampilan pada gambar 4.1.

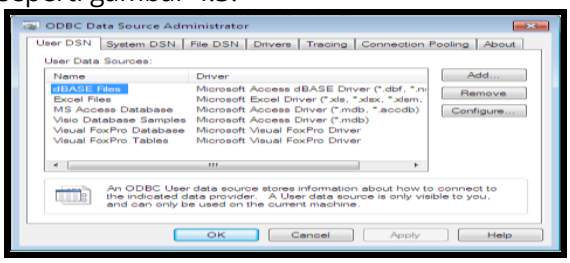
Gambar 14 Membuat Database

Menghubungkan *Database* dengan Form Aplikasi

Menghubungkan database DataNeviMST dilakukan dengan metode ODBC (data source) yang terdiri dari dua tahap yaitu: Membuat *data source* dan membuat kode pada form.

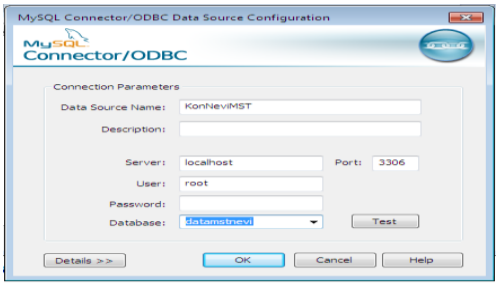
Membuat *Data source*

Untuk membuat *data source* (ODBC), diawali dengan mengaktifkan Control Panel dilanjutkan dengan memilih System and Security ➔ Administrative Tools ➔ Data Source (ODBC) sehingga muncul tampilan seperti gambar 4.3.



Gambar 15 Kotak Dialog ODBC

Selanjutnya memilih tab System DSN, diteruskan menekan tombol Add sehingga muncul dialog Create New Data Source seperti pada gambar 4.4, kemudian dipilih MySql ODBC 5.1 Driver. Setelah menekan tombol finish muncul tampilan konfigurasi *data source*, masukkan nama *data source:* KonNeviMST, server: localhost, user: root; dan database: DataNeviMST, seperti pada gambar.



Gambar 16 Konfigurasi Data Source

Setelah menekan OK, *data source* dengan nama: KonNeviMST sudah berhasil dibuat.

Membuat Kode Koneksi pada Form

Kode koneksi yang digunakan untuk pada setiap form yang akan dihubungkan dengan database adalah sebagai berikut:

```
'Mendeklarasikan variabel con
Dim con As ADODB.Connection
'Prosedur untuk mengatur con
```

e-ISSN 2962-3626
JURNAL KOMPUTER INDONESIA

```
Sub setCon()
Set con = New ADODB.Connection
End Sub
'Memanggil datasource KonNeviMST pada saat form dibuka.
Private Sub Form_Load()
   Me.setCon
   con.ConnectionString = "provider=msdasql.1;" & _
   "persist security info=false; data source=KonNeviMST;"
   con.Open "KonNeviMST"
End Sub
```

Koneksi di atas merupakan koneksi *data source* (ODBC) yang terlebih dahulu dibuat sebelum form, prosesnya ada subbab sebelumnya.

Tampilan Antar Muka (*User Interface*) Aplikasi

Setelah database dan data source dibuat, serta bentuk dari koneksinya sudah ditentukan maka selanjutnya membuat tampilan dari aplikasi menggunakan Visual Basic 6. Penulis tidak akan menjelaskan bagaiman proses pembuatannya namun penjelasan hanya kepada bentuk tampilan *form* aplikasi disertai dengan kode yang ada pada *form* tersebut.

Form Menu Utama

Form yang pertama kali muncul ketika Aplikasi untuk Menentukan *Minimum Spanning Tree* (MST) pada Graf Tak Berarah ini adalah form menu utama, seperti gambar. Form menu utama terdiri dari tiga menu yang menghubungkan menu utama dengan form-form sesuai dengan menu yang di-klik oleh *user*. Ketika user meng-klik menu Simpul diteruskan meng-klik sub menu Input Koordinat Simpul, maka program akan mengaktifkan form Data Simpul, dengan kode sebagai berikut:

1. FormKoordinat.show

Selanjutnya jika *user* meng-klik Pencarian_MST diteruskan Mencari MST maka program akan mengaktifkan form proses dengan kode sebagai berikut:

2. FormProses.show

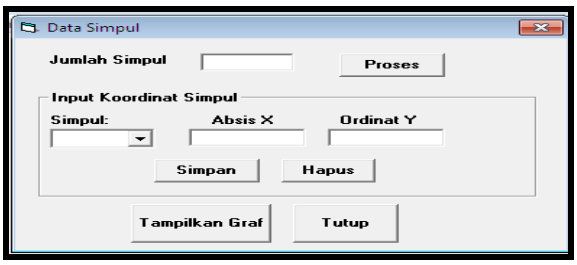
Terakhir, jika *user* meng-klik menu Bantuan maka program akan mengaktifkan form Bantuan, dengan perintah sebagai berikut:

3. FormBantuan.show

Form menu utama juga disertai dengan penjelasan mengenai nama aplikasi, pembuat aplikasi yang disertai tempat bernaungnya.

Form Data Simpul

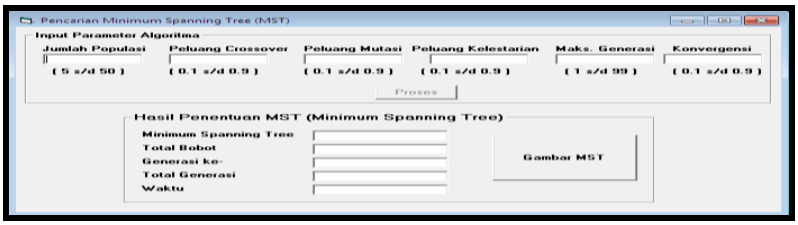
Form data simpul, merupakan form yang digunakan untuk meng-input data simpul (koordinat simpul), lihat gambar 17.



Gambar 11 Form Data Simpul

User harus memasukkan jumlah simpul dan menekan tombol proses agar graf yang akan dicari *minimum spanning tree*-nya tercipta. Tombol proses berfungsi menyimpan data jumlah simpul yang kemudian Form Pencarian *Minimum Spanning Tree* (MST) . Form pencarian MST adalah form

yang digunakan untuk melakukan proses pencarian Minimum Spanning Tree dari graf yang telah dibentuk sebelumnya, adapun tampilan form seperti gambar.

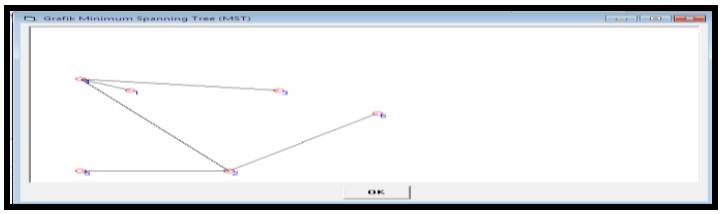


Gambar 12 Form Pencarian Minimum Spanning Tree (MST)

Terlihat pada gambar 4.8 terdapat kotak isian untuk memasukkan data parameter algoritma genetika, artinya untuk melakukan proses penentuan *Minimum Spanning Tree*, *user* harus mengisi kotak-kotak tersebut. Setelah meng-input parameter, selanjutnya adalah menekan tombol proses untuk melakukan proses penentuan *minimum spanning tree* menggunakan algoritma genetika. Adapun kode program tombol tersebut adalah sebagai berikut:

Setelah tombol proses ditekan dan proses berjalan maka hasil dari pencarian / penentuan *minimum spanning tree* (MST) ditampilkan pada kotak di bagian bawah form pencarian.

Minimum spanning tree, yang ditampilkan pada kotak hasil berupa kode *Prufer Number*, namun mengerti atau melihat hasil *minimum spanning tree* dapat dilihat dengan menekan tombol Gambar MST sehingga muncul tampilan graf yang sudah berupa minimum spanning tree, seperti pada gambar 4.9.



Gambar 13 Form Grafik

Pengujian Sistem

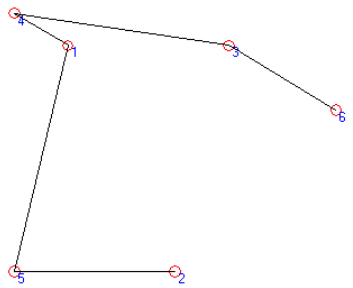
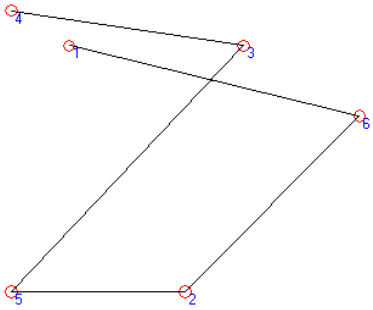
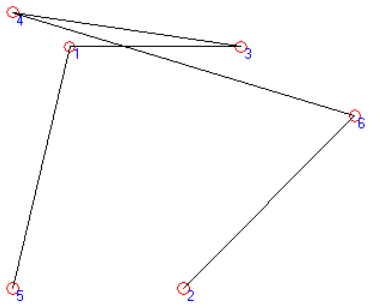
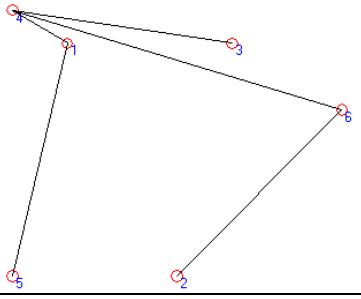
Aplikasi untuk Menentukan Minimum Spanning Tree (MST) pada Graf Tak Berarah dilakukan pengujian dengan metode *Black Box*, yaitu menguji sistem dengan melihat apakah Input yang dimasukkan ke dalam sistem menghasilkan output yang sudah sesuai. Pengujian dilakukan dengan memasukkan enam simpul serta koordinat pada data simpul kemudian diuji dengan memasukkan paramater-parameter algoritma yang berbeda. Adapun aturan dalam memasukkan parameter algoritma seperti yang dijelaskan oleh Kusumadewi & Purnomo (2005; 235), bahwa yang dimaksud dengan parameter di sini adalah parameter kontrol algoritma genetika, yaitu ukuran populasi (popsize), peluang *crossover* (p_c), dan peluang mutasi (p_m). Nilai parameter ini ditentukan juga berdasarkan permasalahan yang dipecahkan. Adapun rekomendasi yang bisa digunakan, antara lain:

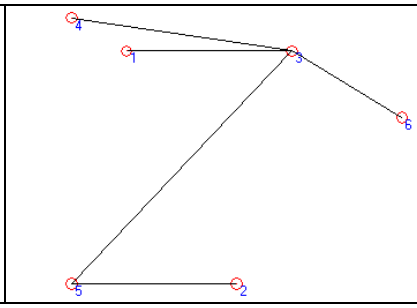
1. Untuk permasalahan yang memiliki kawasan solusi cukup besar, direkomendasikan untuk nilai parameter kontrol:
   (popsize; p_c; p_m) = (50; 0.6; 0.001)
2. Bila rata-rata fitness setiap generasi digunakan sebagai indikator, maka direkomendasikan:
   (popsize; p_c; p_m) = (30; 0.95; 0.01)
3. Bila fitness dari individu terbaik dipantau pada setiap generasi, maka usulannya adalah:
   (popsize; p_c; p_m) = (80; 0.45; 0.01)
4. Ukuran populasi sebaiknya tidak lebih kecil dari 30, untuk sembarang jenis permasalahan.

Sedangkan parameter kelestarian digunakan untuk memantau dan mempertahankan kromosom (individu) yang terbaik pada setiap generasi agar tetap ikut serta regenerasi berikutnya. Parameter maksimum generasi dan konvergensi merupakan parameter untuk kriteria penghentian

proses algoritma genetika. Konvergensi adalah persentase perulangan nilai fitness terbaik dikalikan jumlah generasi maksimum. Misalnya konvergensi =0.2 (20%), maka jika diperoleh individu (kromosom) dengan bobot terkecil pada suatu generasi dan dibandingkan dengan generasi berikutnya sebanyak 20 % dari total generasi masih sebagai yang terkecil atau sama, maka individu tersebut diambil sebagai individu dengan bobot yang optimal. Dengan adanya nilai konvergensi tersebut waktu proses pencarian dengan algoritma genetika akan menjadi lebih singkat. Dengan kata lain tidak harus sampai pada generasi terakhir (generasi maksimum). Berdasarkan aturan tersebut dilakukan pengujian, dengan hasil seperti pada tabel 4.1.

Tabel 1 Hasil Pengujian Program

No	Parameter	Kode MST	Bobot	Waktu (detik)	MST
1	Populasi =35 Crossover =0.5 Mutasi =0.1 Kelestarian =0.1 Maks.Generasi=40 Konvergensi =0.2	5 1 4 3	4.1199998	3.2338723453 3756	
2	Populasi =40 Crossover =0.5 Mutasi =0.2 Kelestarian =0.1 Maks.Generasi=30 Konvergensi =0.2	6 3 5 2	4.1199998	3.1688357629 2905	
3	Populasi =40 Crossover =0.5 Mutasi =0.3 Kelestarian =0.2 Maks.Generasi=60 Konvergensi =0.1	6 1 3 4	4.1199998	5.0556381220 3942	
4	Populasi =35 Crossover =0.4 Mutasi =0.2 Kelestarian =0.1 Maks.Generasi=65 Konvergensi =0.3	6 4 1 4	4.1199998	12.372530821 2435	

5	Populasi =45 Crossover =0.4 Mutasi =0.1 Kelestarian =0.1 Maks.Generasi=70 Konvergensi =0.2	3 5 3 3	4.1199998	11.742242849 1038	

Dari tabel di atas diketahui bahwa bobot MST pada lima kali pengujian tersebut memiliki nilai yang sama dan merupakan nilai terkecil, dengan demikian dapat dikatakan bahwa algoritma genetika berhasil menentukan *minimum spanning tree* (MST) pada graf tak berarah tersebut walaupun jika dilihat dari gambar *minimum spanning tree* yang diperoleh berbeda, namun hal tersebut tidak jadi masalah karena suatu graf dapat saja memiliki lebih dari satu *minimum spanning tree*. Sedangkan jika dilihat dari waktu prosesnya, maka setiap pengujian terlihat ada perbedaan waktu proses. Perbedaan tersebut disebabkan oleh adanya perbedaan pada parameter algoritma yang di-input-kan user terutama generasi maksimum , dan jumlah populasi.

KESIMPULAN DAN SARAN

Kesimpulan

Dari penjelasan dan hasil pengujian dapat disimpulkan bahwa algoritma genetika dapat digunakan untuk menentukan menentukan *Minimum Spanning Tree* (MST) secara optimal. Hasil optimal dari *minimum spanning tree* (MST) yang diperoleh tidak hanya tunggal melainkan dapat berbeda bentuk, sedangkan waktu yang dibutuhkan algoritma genetika dalam menentukan *minimum spanning tree* (MST) tergantung pada parameter yang dimasukkan, semakin tinggi jumlah populasi dan generasi maksimum maka waktu proses akan semakin besar (lama).

Saran

Dalam menggunakan aplikasi untuk menentukan minimum spanning tree ini, hendaknya menggunakan parameter yang sesuai dengan skala yang disarankan sehingga hasil yang diperoleh dapat sesuai dengan harapan *user*. Untuk penelitian selanjutnya, agar penelitian lebih bermanfaat lagi. *Minimum Spanning Tree* (MST) hendaknya diterapkan kepada permasalahan sehari-hari, seperti permasalahan jaringan komputer, jadwal pembangunan atau perbaikan jalan raya.

DAFTAR PUSTAKA

Jogiyanto, 2005, Analisis dan Desain Sistem Informasi; Pendekatan Terstruktur Teori dan Praktek Aplikasi Bisnis, Edisi III, Andi, Yogyakarta.

Monifani, Emsi M.Y. 2014. Pencarian Minimum Spanning Tree (Mst) dengan teknik pengkodean kromosom menggunakan Prufer Sequences pada graf lengkap dengan Algoritma Genetika

Munir, Rinaldi. 2005. Matematika Diskrit. Bandung : Informatika.Bandung.

Pratama, Y.S.Mangontang. 2012. Kompresi pohon dengan kode Prufer Sekolah Teknik Elektro dan Informatika Institut Teknologi Bandung. Bandung.

Sutojo, Edy Mulyanto dan Vincent Suhartono. 2010.Kecerdasan Buatan. Yogyakarta : Penerbit Andi. Yogyakarta