PERBANDINGAN HASIL PERHITUNGAN JARAK TERPENDEK ANTARA ALGORITMA DIJKSTRA DENGAN PEMROGRAMAN LINIER

COMPARISON OF SHORTEST DISTANCE CALCULATION BETWEEN DIJKSTRA'S ALGORITHM AND LINEAR PROGRAMMING

Djoni Dwijono

Program Studi Sistem Informasi, Fakultas Teknologi Informasi
Universitas Kristen Duta Wacana – Yogyakarta 55224
djonid@staff.ukdw.ac.id

Abstrak

Perhitungan jarak terpendek suatu jaringan, misalnya jaringan jalan raya yang menghubungkan satu tempat ke tempat lain, dapat dilakukan dengan berbagai cara. Salah satu cara yang terkenal adalah dengan algoritma. Algoritma yang banyak digunakan adalah Algoritma Dijkstra. Namun, apakah Algoritma Dijkstra memang sudah benar-benar dapat diandalkan untuk menghitung jarak terpendek dari satu tempat ke tempat lain dari suatu jaringan jalan yang memiliki banyak jalur jalan. Untuk mengetahui hal tersebut, hasil perhitungan jarak terpendek dari Algoritma Dijkstra akan dibandingkan dengan cara perhitungan lain, yakni dengan pemrograman linier. Hasil yang diperoleh adalah pemrograman linier mampu menghitung jarak terpendek yang lebih pendek dibandingkan dengan jarak terpendek yang dihitung dengan algoritma Dijkstra.

Kata Kunci: jaringan, Algoritma Dijkstra, pemrograman linier, jarak terpendek

Abstract

Calculating the shortest distance of a network, such as the road network, which connects one place to another, can be performed in various ways. One well-known way is to use algorithm and the most applied one is Dijkstra's algorithm. The problem is whether or not Dijkstra's algorithm is completely reliable to calculate the shortest distance from one place to another on a road network that has a lot of trails. For this purpose, the calculation of the shortest distance applying algorithm Dijkstra will be compared to another calculation method, i.e., linear programming. The finding showed that applying linear programming resulted in the shortest distance than applying the Dijkstra's algorithm.

Keywords: *Network, Dijkstra Algorithm, Linear Programming, Shortest Route*

Tanggal Terima Naskah : 23 Maret 2017
Tanggal Persetujuan Naskah : 23 Mei 2017

1. PENDAHULUAN

Saat ini sudah banyak kendaraan yang memasang alat yang disebut Sistem Penentu Posisi Global (*Global Positioning System*, GPS). Fungsi utama alat tersebut adalah untuk mengetahui koordinat posisi kendaraan pada suatu peta digital Google (*Google Maps*) yang

terdapat di alat tersebut. Selain manfaat tersebut, GPS juga dapat menunjukkan jalur jalan yang ada di peta pada saat tujuan yang diinginkan sudah dimasukkan, sehingga pengendara hanya tinggal mengikuti jalur jalan yang ditunjukkan pada peta digital tersebut, walaupun sangat mungkin pengendara belum mengetahui dengan baik jalur jalan yang akan dilaluinya tersebut. GPS masih memiliki manfaat lain, misalnya informasi kecepatan kendaraan, sejarah perjalanan, cara mematikan mesin kendaraan, memilih jalur jalan alternatif, dan sebagainya. Perkembangan terakhir, seseorang yang memiliki *smartphone* berbasis Android, saat ini juga sudah dapat mengunduh dan memfungsikan GPS pada *smartphone* tersebut, sehingga semua manfaat-manfaat GPS sudah dapat dimanfaatkan melalui *smartphone* miliknya tersebut.

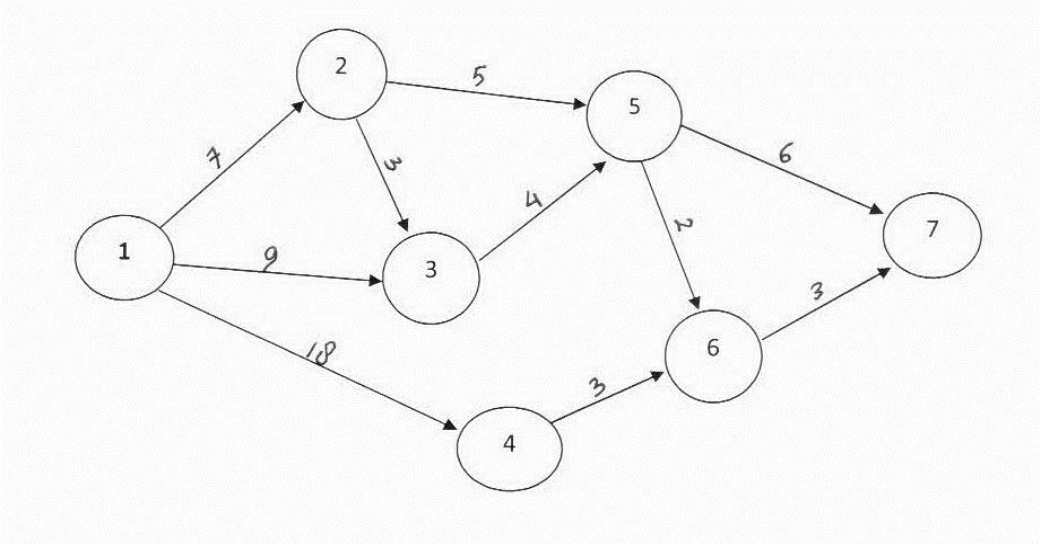
Jalur jalan yang harus dilalui pada GPS tersebut, dari posisi awal kendaraan ke tempat tujuan tentunya sudah diperhitungkan jarak terpendeknya oleh GPS yang ada. Pengendara kendaraan tentunya tidak pernah memikirkan dengan teliti bagaimana GPS dapat menentukan jalur jalan terpendek tersebut. Bagi seorang ahli teknologi informasi, tentunya dia sudah memikirkan dengan baik bagaimana cara menentukan jalur terpendek tersebut [1].

Algoritma untuk menentukan jalur terpendek (*shortest path problem*) yang paling banyak digunakan adalah algoritma Dijkstra. Algoritma lain seperti algoritma Greedy, algoritma A* (*A Star*), algoritma Genetika (*Genetic*), algoritma Bellman-Ford, algoritma Flyod Warshall, algoritma Koloni Semut (*Ant Colony*), algoritma *Branch and Bound*, algoritma Prim, dan sebagainya. Setiap algoritma dapat diimplementasikan dengan berbagai perangkat lunak komputer berupa bahasa pemrograman yang ada, misalnya C, C++, dan dengan implementasi ini maka perhitungan jarak terpendek dapat dilakukan.

Namun, yang menjadi persoalan apakah benar jika yang dipilih adalah algoritma Dijkstra, maka algoritma ini sudah dengan benar menentukan jarak terpendek yang mampu dikerjakannya? Untuk keperluan tersebut maka algoritma Dijkstra akan dibandingkan hasilnya dengan cara lain untuk menghitung jalur terpendek, yakni dengan pemrograman linier [2].

2. JARINGAN JALAN

Berikut merupakan contoh perhitungan jalur terpendek dengan algoritma Dijkstra dan dengan pemrograman linier, dimana di sini akan diberi contoh suatu jaringan yang menunjukkan jalur jalan yang harus dilalui oleh pengendara mobil dari PT ABC yang berangkat dari kantor pusat dan berakhir di pabrik tempat produksi barang-barang milik perusahaan tersebut [3].



Gambar 1. Jaringan jalan antar Kantor Pusat dengan Pabrik PT ABC

Pada Gambar 1, kantor Pusat PT ABC diberi simbol berupa *node* 1 sebagai *node* awal, sedangkan pabrik sebagai tujuan akhirnya diberi simbol *node* 7. Masih ada lima *node* untuk simbol lokasi-lokasi tertentu, yakni *node* 2, 3, 4, 5, dan 6 yang dapat berupa lokasi-lokasi atau gedung-gedung tertentu [4].

Antara satu *node* dengan *node* lain dihubungkan garis anak panah yang merepresentasikan jarak dari satu lokasi menuju lokasi lainnya. Setiap garis tertera angka tertentu, dan angka tersebut dapat dibaca sebagai jarak, misalnya dalam kilometer (km).

3. MENGHITUNG JARAK TERPENDEK DENGAN ALGORITMA DIJKSTRA

Algoritma Dijkstra (*Dijkstra's Algorithms*) diperkenalkan oleh Edsger Wybe Dijkstra (1930 – 2002) pada tahun 1959 di dalam jurnal *Numerische Mathematik* dengan judul "*A Note on Two Problems in Connexion with Graphs*". Algoritma Dijkstra bertujuan untuk menemukan jalur terpendek berdasarkan bobot (jarak) terkecil dari satu titik (*node*) ke titik lainnya yang dimulai dari titik awal dan berakhir pada titik akhir [5].

Algoritma Dijkstra mengimplementasikan teori graf (*graph theory*). Karena itu algoritma Dijkstra dapat dijelaskan dengan memakai simbol *node* (*vertex*, simpul) dengan sisi (*edge*) atau busur (*arc)*, dan karena algoritma ini menggunakan graf berarah atau digraf (*directed graph*) dengan bobot tertentu maka disebut jaringan [6].

Bobot jaringan dalam hal ini adalah jarak dan yang kemudian dipergunakan untuk menentukan dan menghitung jalur terpendeknya. Bobot jaringan dapat digantikan kecepatan kendaraan, kapasitas maksimum jumlah kendaraan, dan berbagai bobot lainnya [7].

Urutan langkah-langkah algoritma Dijkstra adalah sebagai berikut:

1. Berilah nilai bobot (jarak) dari satu *node* ke *node* lainnya, kemudian tentukan nilai 0 pada *node* awal dan nilai tak hingga (*infinite)* pada *node* yang lain atau *node* yang belum terisi
2. Tentukan semua *node* yang belum terjamah dan aturlah *node* awal sebagai *node* keberangkatan
3. Dari *node* keberangkatan, pertimbangkan *node* tetangga yang belum terjamah dan hitung jaraknya dari *node* keberangkatan. Pada urutan ke 3 ini, misalnya *node* A ke *node* B memiliki jarak 7 dan dari *node* B ke *node* C berjarak 10, maka jarak ke *node* C melewati *node* B adalah 7 + 10 = 17. Jika jarak ini lebih kecil dari sebelumnya yang telah direkam, hapus data lama dan simpan ulang jarak dengan jarak yang baru.
4. Pada saat selesai mempertimbangkan setiap jarak terhadap *node* tetangga, tandai *node* yang telah terjamah sebagai *node* terjamah. *Node* terjamah tidak akan pernah diperiksa ulang, dan jarak yang disimpan adalah jarak terakhir dan yang paling minimal jaraknya
5. Tentukan *node* belum terjamah dengan jarak terkecil dari *node* keberangkatan, sebagai *node* keberangkatan selanjutnya, dan lanjutkan dengan kembali ke urutan 3

Untuk menggunakan algoritma Dijkstra tersebut, berikut ini akan dijelaskan langkah per langkah pencarian jalur terpendek yang dimulai dari *node* awal sampai dengan *node* terakhir pada kasus masalah jarak terpendek dari jaringan jalan PT ABC di depan, sehingga jalur terpendek berhasil diperoleh. Urutan langkah-langkah tersebut adalah seperti berikut ini:

1. Antara satu *node* ke *node* lainnya telah diberi bobot berupa jarak. Berilah nilai = 0 pada *node* 1 dan nilai tak hingga pada *node* lainnya.
2. Hitunglah terhadap *node* tetangga (*node* yang belum terjamah) yang terhubung langsung dengan *node* 1 sebagai *node* keberangkatan, dan hasil yang diperoleh adalah *node* 2 karena bobotnya paling kecil dibandingkan bobot pada *node* lainnya. Bobot nilainya = 0 + 7 = 7.

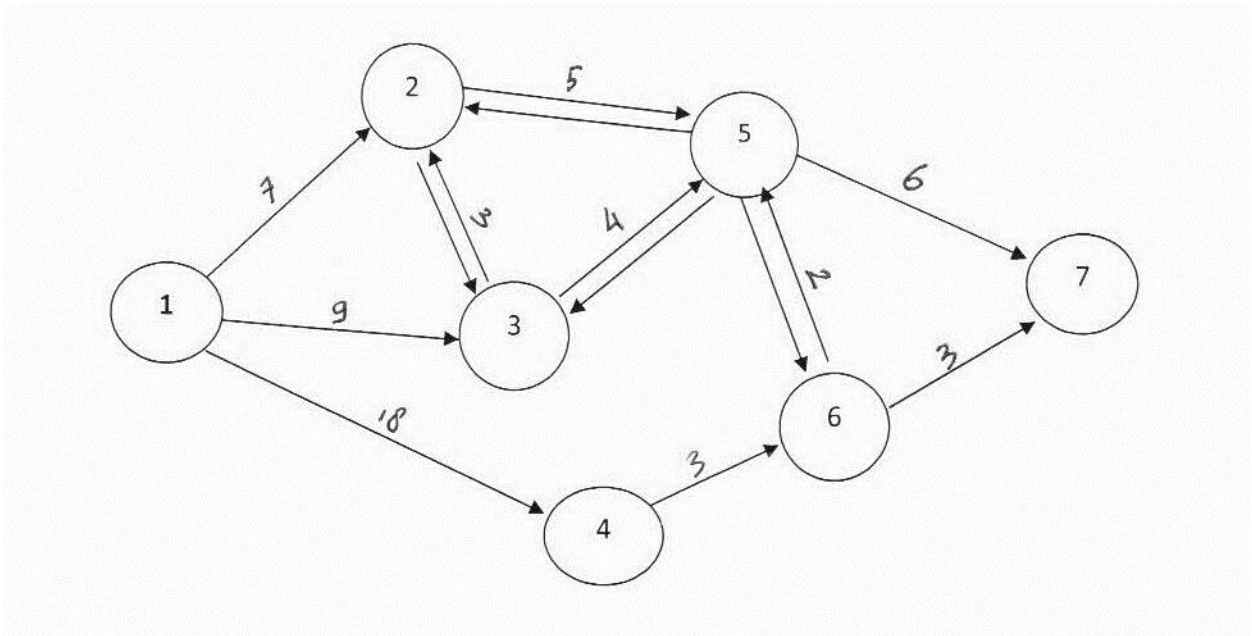
3. Tentukan *node* 2 sebagai *node* keberangkatan dan tandai sebagai *node* yang terjamah. Hitunglah kembali *node* tetangga yang terhubung langsung dengan *node* yang terjamah. Perhitungan menunjukkan *node* 3 menjadi *node* keberangkatan berikutnya karena bobotnya paling kecil dari perhitungan terakhir. Bobotnya adalah = 0 + 9 = 9.
4. Tandai *node* 3 sebagai *node* terjamah. Dari semua *node* tetangga yang belum terjamah, tandailah *node* selanjutnya sebagai *node* yang terjamah, yaitu *node* 5 yang bobotnya = 9 + 4 = 13
5. *Node* 5 sebagai *node* terjamah, selanjutnya lakukan perhitungan kembali. Ditemukan bahwa *node* 7 sebagai *node* tujuan telah tercapai melalui *node* 5, bobotnya 9 + 4 + 6 = 19. *Node* 6 bukan *node* tujuan akhir, walau bobotnya lebih kecil, yakni 9 + 4 + 2 = 15, maka *node* 6 tidak dipilih. Sekarang *node* tujuan telah tercapai dan perhitungan dinyatakan selesai.

Jadi jalur terpendek dari kantor pusat ke gudang milik PT ABC menurut algoritma Dijkstra adalah node 1-3-5-7 = 9 + 4 + 6 = 19 km.

4. MENGHITUNG JARAK TERPENDEK DENGAN PEMROGRAMAN LINIER

Pada pemrograman linier (*linear programming*), yang berada pada bidang Riset Operasi (*Operation Research*) atau Sains Manajemen (*Management Science*), untuk mengembangkan model masalah jalur terpendek adalah memahami masalah tersebut sebagai masalah *transhipment*. Masalah *transhipment* adalah pengembangan masalah transportasi dengan *node* perantara yang disebut *transhipment nodes*, dan *node* ini berfungsi sebagai lokasi perantara yang ditambahkan pada masalah transportasi tersebut [8].

Dalam kasus ini, dengan *node* awal (*node* 1), *node* akhir (*node* 7), dan ada lima *node* perantara (*node* 2, 3, 4, 5, dan 6). Pada Gambar 2, *node transhipment* digambarkan dengan memakai dua garis anak panah yang berlawanan arah atau *node* yang memiliki jalur keluar dan jalur masuk, hanya khusus untuk *node* 4 memiliki satu jalur masuk dan 1 jalur keluar, karena tidak memungkinkan ada jalur yang masuk dari *node* berikutnya. *Node* awal dan *node* akhir jelas bukan *node transhipment* karena tidak memiliki dua jalur yang masuk dan keluar.



Gambar 2. Jaringan *transhipment* untuk masalah jalur terpendek PTABC

Untuk menggambarkan hubungan dari satu *node* ke *node* berikutnya, digunakan simbol seperti berikut ini:

X_{ij}= jalur dari *node* i ke *node* j

Untuk mendapatkan jalur terpendek antara *node* 1 ke *node* 7, dianggap *node* 1 memiliki *supply* = 1 dan *node* 7 dianggap memiliki *demand* = 1, misalnya X_{ij} menggambarkan jumlah arus unit yang dikirim dari *node* i ke *node* j. Karena dianggap hanya satu unit yang akan dikirim dari *node* 1 ke *node* 7, maka nilai dari X_{ij} hanya 0 atau 1. Jadi jika X_{ij}= 1, maka ini adalah jalur terpendek, sedangkan jikaX_{ij}= 0, jalur ini bukan jalur pendek.

Dengan pemrograman linier yang disusun untuk menggambarkan jarak terpendek dari *node* 1 ke *node* 7, maka fungsi objektif untuk PT.ABC adalah:

Minimumkan $7X_{12} + 9X_{13}+ 18X_{14}+ 3X_{23}+ 3X_{32}+ 5X_{25}+ 5X_{52}+ 4X_{35}+ 4X_{53}+ 3X_{46}+ 2X_{56}+ 2X_{65}+ 6X_{57}+ 3X_{67}$

Untuk menentukan batasan *node* 1 yang dianggap memiliki *supply* = 1, maka arus keluar dari *node* 1 harus sama dengan 1, maka batasannya adalah:

$X_{12}+ X_{13}+ X_{14} = 1$

Untuk *transhipment* di *node* 2, 3, 4, 5, dan 6, maka arus keluar harus sama dengan arus masuknya, maka hasilnya = 0. Batasan-batasan untuk *node* tersebut adalah:

	Arus Keluar	Arus Masuk	
Node 2	$X_{23}+ X_{25}$	$-X_{12}- X_{32}- X_{52}$	= 0
Node 3	$X_{32}+ X_{35}$	$-X_{13}- X_{23}- X_{53}$	= 0
Node 4	X_{46}	$-X_{14}$	= 0
Node 5	$X_{57}+ X_{56}$	$-X_{25}- X_{35}- X_{65}$	= 0
Node 6	$X_{67}+ X_{65}$	$-X_{46}- X_{56}$	= 0
			= 0

Terakhir adalah batasan untuk *node* 7 = 1, karena *node* tersebut adalah *node* tujuan yang memiliki *demand* = 1, maka batasan tersebut adalah:

$X_{57}+ X_{67} = 1$

Jadi pemrograman linier yang diperoleh 14 variabel dan 7 batasan, dan jika ditulis secara urut akan tampak seperti berikut ini:

Min $7X_{12} + 9X_{13}+ 18X_{14}+ 3X_{23}+ 3X_{32}+ 5X_{25}+ 5X_{52}+ 4X_{35}+ 4X_{53}+ 3X_{46}+ 2X_{56}+ 2X_{65} + 6X_{57}+ 3X_{67}$

Batasan

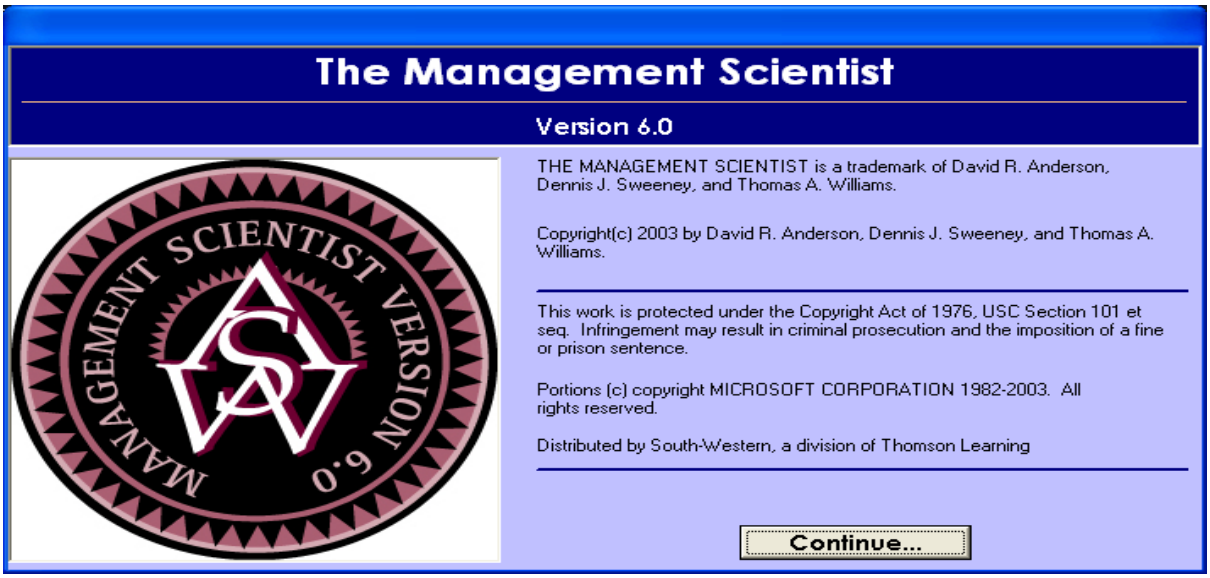
$X_{12}+ X_{13}+ X_{14} = 1$
$-X_{12} + X_{23}+X_{25}- X_{32}- X_{52} = 0$
$-X_{13}-X_{23} +X_{32} + X_{35}-X_{53} = 0$
$-X_{14} + X_{46} = 0$
$- X_{25} - X_{35} + X_{56}+ X_{57} - X_{65} = 0$
$-X_{46} -X_{56}+ X_{65} + X_{67} = 0$
$X_{57}+ X_{67} = 1$

$X_{ij} \geq 0$ untuk semua i dan j

Jika pemrograman linier sudah siap maka selanjutnya diproses menggunakan perangkat lunak *Management Scientist Version* 6.0. Prosedur untuk mengerjakannya sebagai berikut:

1. Panggil Perangkat Lunak *Management Scientist*

Memanggil perangkat lunak ini dengan mengklik dua kali pada berkas MS60.EXE pada *folder*-nya, maka akan tampak tampilan seperti berikut:

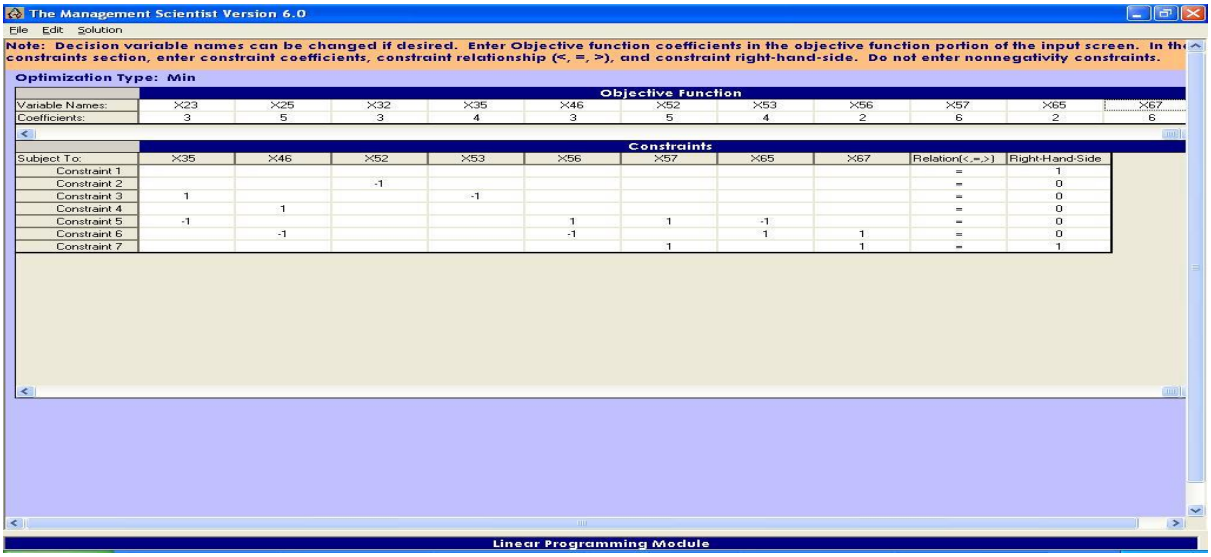


Gambar 3. Tampilan perangkat lunak *Management Scientist Version* 6.0

Sesudah tampilan tersebut tampak di layar, maka lakukan langkah-langkah berikut ini:

2. Pilih *Continue*
3. Pilih *Linear Programming*
4. Masukkan 14 pada *Variables*
5. Masukkan 7 pada *Constraints*
6. Pilih Minimum
7. Pilih Ok
8. Pilih *File*
9. Pilih *New*
10. Masukkan data dari pemrograman linier pada menu isian yang tampil.

Jangan lupa mengganti *Variable Names* dengan nama-nama variabel yang ada pada pemrograman linier, yakni X12 untuk X_{12} dan seterusnya. Jika sudah diisi lengkap akan tampak seperti Gambar 4 berikut ini:

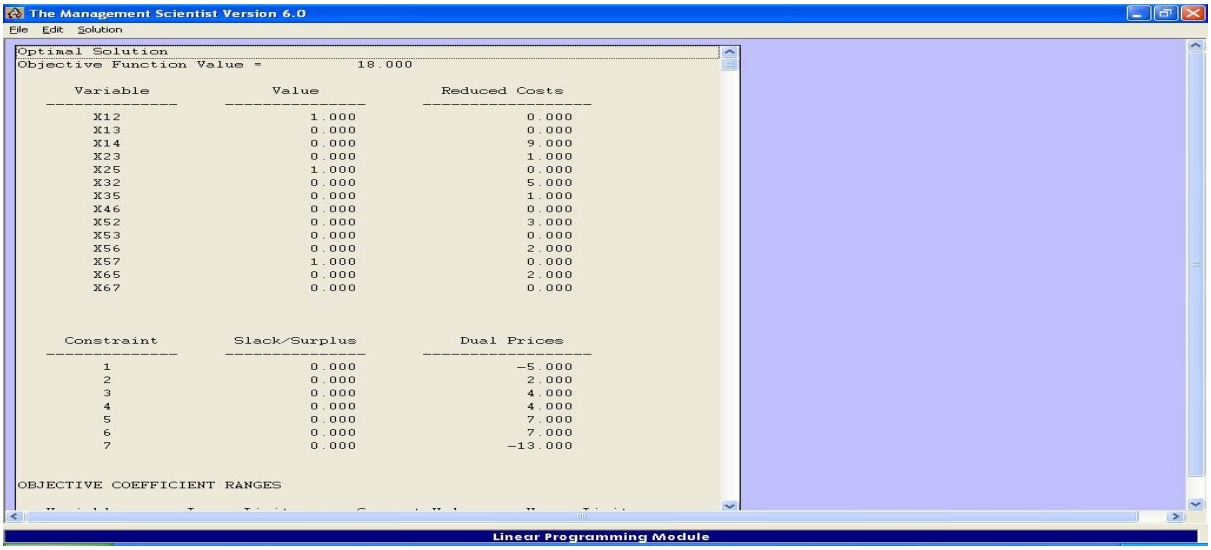


Gambar 4. Tampilan data pengisian untuk pemrograman linier jarak terpendek PT ABC

Jika sudah terisi lengkap, periksalah letak dan isi datanya dengan dengan benar dan jika sudah yakin benar, maka lakukan langkah berikut:

11. Pilih *Solution*
12. Pilih *Solve*

Maka akan tampak hasilnya seperti pada Gambar 5 berikut ini:



Gambar 5. Hasil pemrosesan pemrograman linier jarak terpendek PT ABC

Jadi jarak terpendek berdasarkan perhitungan pemrograman linier yang diproses dengan *Management Scientist Version* 6.0 adalah jalur-jalur berikut:

$X_{12} = 1$, $X_{25} = 1$ dan $X_{57} = 1$

Rute terpendeknya dari *node* 1 ke *node* 7 adalah 1-2-5-7 atau *node* 1 ke *node* 2, dari *node* 2 ke *node* 5 dan terakhir dari *node* 5 ke *node* 7. Dengan kata lain, dari hasil perhitungan pemrograman linear, maka jarak terpendek tersebut adalah:

$X_{12}+ X_{25}+ X_{57} = 7 + 5 + 6 = 18$ Km

5. PERBANDINGAN HASIL PERHITUNGAN

Hasil perhitungan jalur terpendek dengan:

Algoritma Dijkstra = 19 Km
Pemrograman Linier = 18 Km

Jadi hasilnya adalah:

Jalur terpendek pemrograman linier < jalur terpendek algoritma Dijkstra

6. KESIMPULAN

Beberapa kesimpulan dapat diambil dari analisis yang telah dilakukan, yakni:

1. Hasil perhitungan jalur terpendek antara algoritma Dijkstra dengan pemrograman linier ternyata berbeda. Hasil perhitungan jarak terpendek dengan pemrograman linier lebih kecil dibandingkan dengan algoritma Dijkstra, dengan demikian pemrograman linier lebih baik dari algoritma Dijkstra dalam menghitung jalur pendek
2. Algoritma Dijkstra dapat dikerjakan secara manual atau diimplementasikan dengan berbagai bahasa pemrograman sedangkan pemrograman linier tidak bisa diimplementasikan dengan bahasa pemrograman, karena pemrograman linier harus dibuat terlebih dahulu secara manual, baru kemudian prosesnya dapat dikerjakan dengan program jadi yang khusus untuk memproses pemrograman linier tersebut

REFERENSI

[1]. Anderson, David R., & Sweeney, Dennis J., & Williams, Thomas A., & Camm, Jeffrey D., & Cochran, James J., & Fry, Michael J., & Chimann, Jeffrey W. 2013. "Quantitative Methods for Business, 12th Edition". South-Western.

[2]. Anderson, David R., & Sweeney, Dennis J., & Williams, Thomas A., & Martin, Kipp. 2008. "An Introduction to Management Science, Quantitative Approaches to Decision Making, 12th Edition". Thomson South-Western.

[3]. Conradie, Willem., & Goranko, Valentin. 2015. "Logic and Discrete Mathematics, 1st Published". John and Wiley.

[4]. Cormen, Thomas H., & Leiserson, Charles E. 2009. "Introduction to Algorithms, 3rd Edition". The MIT Press.

[5]. Goodaire, Edgar D., & Parmenter, Michael M. 2002. "Discrete Mathematics with Graph Theory, 2nd Edition". Prentice Hall.

[6]. Epp, Susanna S. 2011. "Discrete Mathematics: Introduction to Mathematical Reasoning, 4th Edition". Brooks/Cole.

[7]. Rosen, Kenneth H. 2011. "Discrete Mathematics and Its Application, 7th Edition". McGraw Hill Education.

[8]. Tan, Soo.T. 2009. "Finite Mathematics: For The Managerial, Life and Social Science, 9th Edition". Brook/Cole.