Jurnal Matematika UNAND
Vol. **VI** No. **4** Hal. 1 – 8
ISSN : 2303–291X

ALGORITMA DIJKSTRA : TEORI DAN APLIKASINYA

ABDUL ZAKI

Program Studi Magister Matematika,
Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Andalas,
Kampus UNAND Limau Manis Padang, Indonesia,
email : zakimathua@yahoo.com

Abstrak. Algoritma *Dijkstra* merupakan algoritma yang dipakai dalam penentuan lintasan terpendek dari suatu titik tertentu ke setiap titik lain pada suatu graf. Lintasan terpendek untuk suatu titik tertentu dengan titik lainnya diperoleh dari pohon pembangun yang memiliki nilai minimum. Pada makalah ini akan dibahas teori dari algoritma *Dijkstra* serta penerapannya dalam menentukan lintasan terpendek.

Kata Kunci: Pohon pembangun, algoritma Dijkstra, lintasan terpendek

1. Pendahuluan

Teori graf adalah bagian dari disiplin ilmu matematika diskrit yang sangat berguna untuk mengembangkan model-model terstruktur dalam berbagai situasi dan sangat banyak diaplikasikan dalam kehidupan sehari-hari. Titik dan garis merupakan alat yang dipakai dalam teori graf. Sebagai contoh, titik merupakan representasi dari orang-orang pada suatu keluarga dan garis merupakan representasi dari hubungan antara dua orang dalam keluarga tersebut. Salah satu aplikasi teori graf adalah penentuan lintasan terpendek pada suatu graf. Aplikasi penentuan lintasan terpendek merupakan salah satu persoalan optimasi karena tujuannya adalah untuk menentukan panjang lintasan yang minimum dari satu titik ke titik lain.

Graf yang dipakai dalam penentuan lintasan terpendek adalah graf berbobot, yaitu graf yang untuk setiap sisinya memiliki nilai sehingga panjang lintasan dari suatu titik ke titik lain merupakan jumlah dari nilai-nilai pada setiap sisinya. Panjang lintasan pada lintasan terpendek dari suatu titik ke titik lain dapat disebut dengan jarak antara dua titik.

Penentuan lintasan terpendek yang akan dibahas pada makalah ini menggunakan algoritma *Dijkstra* yang dikembangkan oleh Edsger Wybe Dijkstra pada tahun 1959. Algoritma *Dijkstra* merupakan algoritma yang dipakai dalam penentuan lintasan terpendek dari suatu titik tertentu ke setiap titik lain pada suatu graf. Pengaplikasian algoritma *Dijkstra* dalam menentukan lintasan terpendek bertujuan agar lintasan terpendek dari suatu titik ke titik lain dapat dilakukan dalam satu kali algoritma saja. Konsep dari algoritma *Dijkstra* adalah membentuk pohon *Dijkstra* pada suatu graf G terhubung dengan pengulangan sebanyak $|V(G)| - 1$ sedemikian sehingga pohon *Dijkstra* setelah pengulangan terakhir merupakan pohon pembangun (*spanning tree*) dari graf G.

2. Definisi dan Terminologi Graf

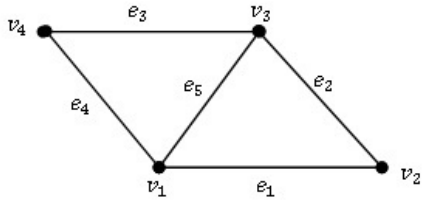
Berikut disajikan definisi dan terminologi dalam teori graf yang digunakan pada makalah ini, yang dikutip dari [1].

Suatu graf $G = (V, E)$ didefinisikan sebagai suatu pasangan himpunan $(V(G), E(G), \psi_G)$, di mana $V(G)$ adalah himpunan tak kosong dari elemen yang disebut **titik** (*vertex*), $E(G)$ adalah himpunan **sisi** (*edge*) dan ψ_G adalah suatu fungsi keterkaitan (*incidency function*) yang mengaitkan setiap sisi di G dengan pasangan titik di G. Banyaknya titik di G disebut **orde** (*order*) dari G yang dinotasikan dengan $|V(G)| = P$, dan banyaknya sisi pada G disebut **ukuran** (*size*) dari G yang dinotasikan dengan $|E(G)| = Q$.

Misal terdapat graf $G = (V, E)$ dengan $v_i, v_j \in V(G)$. Jika untuk setiap dua titik $v_i, v_j \in V(G)$ terdapat lintasan yang menghubungkan kedua titik tersebut, maka graf G dikatakan **graf terhubung (*connected graph*)**.

Graf yang digunakan untuk menentukan lintasan terpendek adalah graf terhubung. Lintasan (*path*) merupakan hasil yang akan diperoleh dari penelitian ini dengan memperhatikan bahwa titik awal (v_0) tidak sama dengan titik akhir (v_k).

Suatu barisan yang memuat titik-titik dan sisi-sisi bergantian $W = v_0e_1v_1e_2 \dots e_kv_k$ pada graf terhubung G dinamakan **jalan (*walk*)**. Misalkan $W = v_0e_1v_1 \dots e_kv_k$ adalah jalan, untuk setiap sisi-sisi $e_1, e_2, \dots, e_k$ dan titik-titik $v_0, v_1, \dots, v_k$. Jika $e_i \neq e_j$, $i, j = 1, 2, \dots, k$ dan $v_s \neq v_t$, $s, t = 0, 1, \dots, k$, maka W adalah **lintasan-(v_0, v_k)**. Dapat dilihat bahwa lintasan merupakan jalan yang setiap titik dan sisinya hanya dilewati satu kali.



Gambar 1. Ilustrasi Jalan dan Lintasan pada Graf G

Graf G pada Gambar 1 memiliki jalan yang dinotasikan dengan W dan lintasan yang dinotasikan dengan P sebagai berikut:

$$W = v_1e_5v_3e_2v_2e_1v_1e_5v_3e_3v_4e_4v_1,$$
$$P = v_1e_1v_2e_2v_3e_3v_4.$$

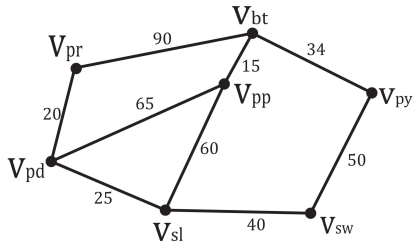
2.1. *Graf Berbobot*

Graf berbobot sangat banyak digunakan dalam aplikasi teori graf. Secara sederhana, graf berbobot adalah graf yang setiap sisinya diberikan suatu nilai atau bobot. Nilai atau bobot tersebut dapat bernilai positif atau negatif, akan tetapi pada kasus lintasan terpendek, bobot harus bernilai positif karena berhubungan dengan panjang lintasan [1].

Definisi 2.1. *[1] Pengaitan suatu bilangan riil $w(e)$ untuk setiap sisi e di graf G disebut* ***bobot (weighted)****. Graf G yang memiliki nilai $w(e)$ pada sisi-sisinya dinamakan* ***graf berbobot (weighted graph)****.*

.

Sebagai contoh, misalkan kota-kota di Sumatera Barat direpresentasikan sebagai titik-titik pada graf S, yaitu Padang (v_{pd}), Bukittinggi (v_{bt}), Padang Panjang (v_{pp}), Payakumbuh (v_{py}), Pariaman (v_{pr}), Solok (v_{sl}) dan Sawahlunto (v_{sw}), seperti yang ditampilkan pada Gambar 2. Di sini bobot pada graf S didefinisikan sebagai jarak tempuh antara kedua kota.



Gambar 2. Graf S yang Merupakan Graf Berbobot.

Pada Gambar 2, terdapat pengaitan suatu bilangan real $w(e)$ untuk setiap sisi pada graf S. Pada graf S didefinisikan bahwa bobot untuk $w(v_{pd}v_{sl}) = 25$ artinya jarak antara kota Padang dan kota Solok adalah 25. Hal serupa juga berlaku untuk setiap titik yang terhubung lainnya.

Definisi 2.2. *[1]* ***Panjang lintasan pada graf berbobot*** *adalah jumlah bobot pada suatu lintasan$-(u, v)$ yang dinotasikan dengan $l(u, v)$. Misalkan $v_1v_2 \dots v_n$ adalah suatu lintasan dengan titik awal v_1 dan titik akhir v_n, maka panjang lintasan$-(v_1, v_n)$ dinyatakan oleh*

$$l(v_1, v_n) = \sum_{i=1,2,\dots,n-1} w(v_i v_{i+1}). \tag{2.1}$$

Pada Gambar 2, terdapat banyak kemungkinan lintasan antara Padang dan Payakumbuh. Misalkan $P(v_{pd}, v_{py}) = v_{pd}v_{sl}v_{sw}v_{py}$, maka berdasarkan Persamaan (2.1), panjang lintasan P adalah

$$\begin{aligned} l(v_{pd}, v_{py}) &= w(v_{pd}v_{sl}) + w(v_{sl}v_{sw}) + w(v_{sw}vpy) \\ &= 25 + 40 + 50 \\ &= 115. \end{aligned}$$

2.2. *Pohon Pembangun (Spanning Tree)*

Secara sederhana pohon (*tree*) dari suatu graf merupakan subgraf yang tidak memuat sebuah siklus (*cycle* dan pohon pembangun *spanning tree* merupakan subgraf dari pohon tersebut [3].

Definisi 2.3. *[3] Suatu graf yang tidak memuat siklus dikatakan* ***acyclic****.* ***Hutan (forest)*** *merupakan graf acyclic.* ***Pohon*** *adalah graf acyclic terhubung.* ***Daun (leaf)*** *merupakan titik (vertex) yang berderajat 1.* ***Subgraf pembangun (spanning subgraph)*** *dari G merupakan suatu subgraf yang titik-titiknya adalah $V(G)$. Suatu* ***pohon pembangun*** *merupakan subgraf pembangun yang berbentuk pohon.*

Bobot setiap sisi sangat diperlukan dalam penentuan nilai (*cost*) antara dua titik tertentu pada suatu graf. Pohon pembangun dari suatu graf terhubung yang memiliki bobot total minimum dapat dikatakan sebagai pohon pembangun minimum. Pohon pembangun minimum ini dapat digunakan untuk menentukan lintasan terpendek dari suatu titik tertentu ke titik lainnya pada graf [3].

2.3. *Lintasan Terpendek*

Penentuan lintasan terpendek pada sebuah graf merupakan salah satu persoalan optimasi. Graf yang digunakan dalam penentuan lintasan terpendek adalah graf berbobot.

Definisi 2.4. *[1]* ***Lintasan terpendek (shortest path)*** *dengan titik awal v_0 dan titik akhir v_k didefinisikan sebagai lintasan yang memiliki panjang lintasan yang minimum dari v_0 ke v_k.*

Misalkan v_0 dinyatakan sebagai titik awal dan v_k sebagai titik akhir. Jika terdapat m lintasan yang berbeda, maka lintasan terpendek antara v_0 dan v_k dinyatakan oleh P_t, dimana P_t adalah lintasan ke$-t$ dari v_0 ke v_k, dengan

$$l_t(v_0, v_k) = \min_{i=1,\ldots,m} l_i(v_0, v_k).$$

Pada Gambar 2, dapat dilihat bahwa terdapat enam buah lintasan dari v_{pd} ke v_{py} dengan panjang lintasan yang berbeda. Lintasan terpendek dari titik v_{pd} ke v_{py} adalah $v_{pd}v_{pp}v_{bt}v_{py}$ dengan jarak $d(v_{pd}, v_{py}) = 114$. Pada kasus ini dimungkinkan terdapat dua atau lebih lintasan terpendek dengan panjang lintasan yang sama tetapi jalurnya berbeda. Dengan memperhatikan lintasan terpendek dari v_{pp} ke v_sw, maka dapat dilihat bahwa lintasan $v_{pp}v_{bt}v_{py}v_{sw}$ mempunyai jarak 100 dan lintasan $v_{pp}v_{sl}v_{sw}$ juga mempunyai jarak 100. Jarak dari kedua lintasan tersebut sama, tetapi lintasan yang dilalui berbeda. Pemilihan lintasan terpendek dari v_{pp} ke v_{sw} dilakukan dengan memilih salah satu [2].

Definisi 2.5. *[1]* ***Jarak (distance) pada graf berbobot*** *adalah jumlah bobot minimum pada suatu lintasan-(u, v), dinotasikan dengan $d(u, v)$.*

Berdasarkan definisi panjang lintasan dan Definisi 2.5, jarak pada suatu graf berbobot dapat dikatakan sebagai panjang lintasan dari lintasan terpendek.

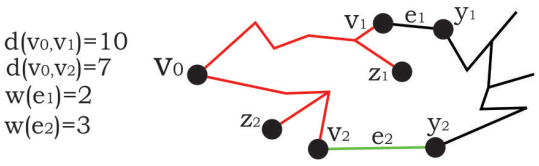
3. Algoritma *Dijkstra*

Algoritma *Dijkstra* merupakan algoritma yang dipakai dalam penentuan lintasan terpendek dari suatu titik tertentu ke setiap titik lain pada suatu graf. Algoritma ini dikembangkan oleh Edsger Wybe Dijkstra pada tahun 1959. Lintasan terpendek

untuk suatu titik tertentu dengan titik lainnya diperoleh dari pohon pembangun yang memiliki nilai minimum [2].

Strategi yang digunakan pada algoritma *Dijkstra* yaitu dengan membentuk suatu pohon *Dijkstra* yang diawali pada titik v_0, dengan menambahkan sisi terkait dan terdekat dengan titik v_0. Penambahan sisi dilakukan pada setiap pengulangan (*iteration*) dan dilakukan dengan menggunakan fungsi *Dijkstra-nextEdge* [2].

Definisi 3.1. *[2] Fungsi* ***Dijkstra-nextEdge*** *didefinisikan sebagai berikut. Misalkan X adalah himpunan sisi terkait yang telah ditambahkan. Dijkstra-nextEdge(G, X) memberikan nilai pada titik akhir sisi terkait yang titik ujungnya berderajat lebih dari satu (non-tree) dan memilih sisi yang titik ujungnya paling dekat dengan v_0. Jika titik yang terpilih lebih dari satu, maka pilih salah satu.*



Gambar 3. Ilustrasi Fungsi *Dijkstra-nextEdge*(G, X)

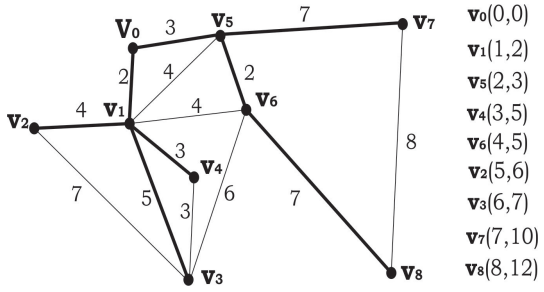
Gambar 3 memperlihatkan bahwa e_1 dan e_2 merupakan sisi terkait yang titik ujungnya berderajat lebih dari satu. Kemudian sisi yang titik ujungnya paling dekat dengan v_0 adalah sisi e_2.

Algoritma Lintasan Terpendek Dijkstra [2]
Input : Graf berbobot G dan titik awal v_0
Output : Pohon lintasan terpendek T dengan titik awal v_0

- Inisialisasikan titik v_0 sebagai pohon T.
- Inisialisasikan X sebagai himpunan sisi yang bertetangga dengan v_0
- Selama $X \neq \emptyset$
  - Misalkan $e :=$ *Dijkstra-nextEdge*(G, X)
  - Misalkan v_1 sebagai titik ujung dari sisi e yang berderajat lebih dari satu.
  - Tambahkan sisi e dan titik v_1 pada pohon T
  - Perbaharui pohon T
- Diperoleh pohon T yang merupakan pohon pembangun dari G

Misalkan $d(v_0, v)$ menyatakan jarak dari titik awal v_0 ke v untuk setiap titik v pada pohon. Gambar 4 memperlihatkan penambahan sisi pada setiap pengulangan untuk membentuk pohon *Dijkstra*.

Angka pertama di dalam kurung menyatakan pengulangan dan angka kedua menyatakan $d(v_0, v)$ untuk setiap $v \in V(G)$. Pada pengulangan ke-1, sisi terkait yang titik ujungnya berderajat lebih dari satu adalah v_0v_1 dan v_0v_5. Titik v_1 memiliki nilai yang lebih rendah dari titik v_5 sehingga sisi yang ditambahkan adalah sisi v_0v_1.



Gambar 4. Proses Penentuan Pohon *Dijkstra* pada Graf G

Pada pengulangan ke-2, sisi terkait yang titik ujungnya berderajat lebih dari satu adalah $v_0v_5, v_1v_2, v_1v_3, v_1v_4, v_1v_6, v_1v_5$. Sisi yang titik ujungnya paling dekat dengan v_0 adalah sisi v_1v_5 sehingga sisi tersebut ditambahkan ke pohon *Dijkstra*. Hal tersebut dilakukan untuk pengulangan selanjutnya sampai pohon *Dijkstra* yang terbentuk merupakan pohon pembangun dari graf G. Pada Gambar 4, sisi-sisi tebal adalah sisi-sisi dari lintasan terpendek dari titik v_0 ke titik lain di graf G yang diperoleh dengan algoritma *Dijkstra*.

3.1. *Penghitungan Jarak*

Misalkan $w(e)$ menyatakan bobot dari sisi e pada graf berbobot dan $w(q)$ menyatakan bobot dari sisi q pada pohon yang terbentuk dan v_0 adalah titik awal yang dipilih untuk penentuan pohon *Dijkstra*. Jika v adalah titik pangkal dengan nilai terendah dan y sebagai titik ujung pada sisi q yang terpilih, maka $d(v_0, y) = d(v_0, v) + w(q)$. Jadi, pada saat q terpilih pada pengulangan ke$-i$ sebagai sisi yang ditambahkan pada pohon *Dijkstra*, y haruslah menjadi titik dengan nilai minimum.

Definisi 3.2. *[2] Misalkan e sebagai sisi terakhir yang ditambahkan pada pohon Dijkstra yang terbentuk dan v merupakan titik pangkal pada sisi pohon e. P-**Value** dari sisi e yang dinotasikan oleh $P(e)$, diberikan oleh*

$$P(e) = d(v_0, v) + w(e) \tag{3.1}$$

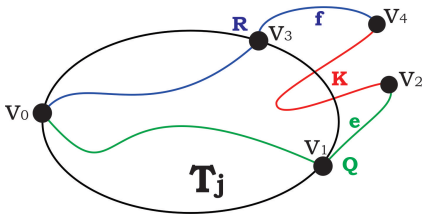
Jadi, *Dijkstra-nextEdge*(G, X) memilih dan memberikan nilai pada titik akhir sisi e sedemikian sehingga $P(e^\star) = \min_{e \in X} P(e)$. Dengan kata lain $e^\star$ merupakan sisi yang memiliki *P-value* yang minimum untuk sisi $e \in X$, dimana X adalah himpunan sisi yang terkait dengan pohon T. *P-value* untuk setiap sisi e dapat berubah pada setiap pengulangan.

3.2. *Uji Kebenaran Algoritma Dijkstra*

Teorema 3.3. *[2] Misalkan T_j adalah pohon Dijkstra setelah j iterasi dari algoritma Dijkstra pada graf terhubung G, untuk $0 \leq j \leq |V(G)| - 1$. Untuk setiap $v \in T_j$, lintasan-(v_0, v) yang unik di T_j adalah lintasan-(v_0, v) terpendek di G.*

Bukti. Solusi trivial untuk T_0, karena pada saat $j = 0$, pohon *Dijkstra* yang terbentuk hanya sebuah titik awal. Menggunakan induksi, diasumsikan untuk setiap j, $0 \leq j \leq |V(G)| - 2$, bahwa T_j adalah benar. Misalkan sisi e dengan titik pangkal $v_1 \in T_j$ dan titik ujung $v_2 \notin T_j$, sebagai sisi terkait yang ditambahkan ke pohon T_j pada pengulangan ke $j + 1$. Karena v_2 merupakan satu-satunya titik baru di T_{j+1}, maka lintasan-(v_0, v_2) Q di T_{j+1} adalah lintasan terpendek dari v_0 ke v_2 dengan panjang lintasan Q adalah $P(e)$.

Selanjutnya akan ditunjukkan bahwa lintasan Q merupakan lintasan yang unik. Artinya, tidak ada lintasan yang lebih pendek dari Q. Misalkan R adalah lintasan-(v_0, v_2) yang lain di G. Notasikan panjang lintasan Q dengan $l(Q)$ dan panjang lintasan R dengan $l(R)$. Akan ditunjukkan $l(R) \geq l(Q)$. Misalkan sisi f dengan titik pangkal $v_3 \in T_j$ dan titik ujung $v_4 \notin T_j$, dimana $f \notin T_j$. Misalkan lintasan K adalah lintasan yang menghubungkan v_4 dengan v_2, artinya lintasan K adalah bagian dari lintasan R.



Gambar 5. Uji kebenaran Algoritma *Dijkstra*

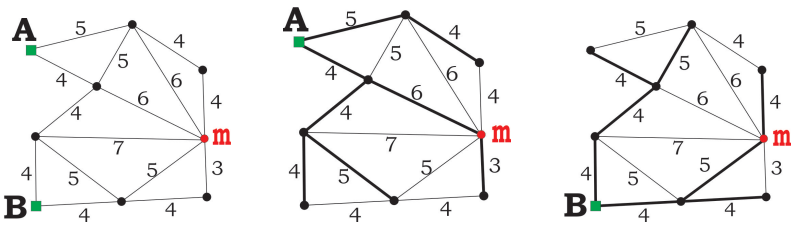
Karena sisi e sisi terkait dengan T_j dan dipilih untuk ditambahkan ke T_j pada pengulangan ke $j + 1$, maka $P(e) \leq P(f)$. Perhatikan bahwa,

$$\begin{aligned} l(R) = d(v_0, v_3) + w(f) + l(K) &= P(f) + l(K) \\ &\geq P(e) + l(K) \geq P(e) = l(Q). \end{aligned}$$

Jadi, karena $l(R) \geq l(Q)$ maka lintasan Q merupakan lintasan unik yaitu tidak ada lintasan yang lebih pendek dari lintasan Q. □

4. Aplikasi Algoritma *Dijkstra*

Algoritma *Dijkstra* dapat digunakan dalam menentukan lintasan terpendek dari satu titik tertentu ke setiap titik lain pada graf yang dibentuk. Banyak hal yang dapat diaplikasikan menggunakan algoritma *Dijkstra*, salah satunya adalah pemilihan lokasi terdekat diantara beberapa lokasi. Hal ini dilakukan dengan mengatur lokasi-lokasi tersebut sebagai titik awal dalam proses pembentukan pohon *Dijkstra*. Misalkan dua lokasi, yaitu lokasi A dan lokasi B ditempatkan sebagai titik yang akan dituju oleh setiap individu (Gambar 6). Selanjutnya individu m ataupun individu lain yang ditempatkan pada titik di graf dapat memilih lintasan terpendek dari dua pohon *Dijkstra* yang terbentuk. Kedua pohon *Dijkstra* tersebut dibangun dari titik awal A dan titik awal B.



Gambar 6. Graf G, Pohon *Dijkstra* Lokasi A dan Pohon *Dijkstra* Lokasi B

Bobot pada graf G menyatakan jarak diantara dua titik. Lokasi terdekat untuk individu m adalah lokasi B, karena $d(B, m) < d(A, m)$. Hal tersebut berlaku untuk individu lain yang ditempatkan pada titik di graf G.

5. Kesimpulan dan Saran

Adapun kesimpulan dari makalah ini dapat dijelaskan sebagai berikut

(1) Algoritma *Dijkstra* merupakan algoritma yang dipakai dalam penentuan lintasan terpendek dari suatu titik tertentu ke setiap titik lain pada suatu graf.
(2) Strategi yang digunakan pada algoritma *Dijkstra* yaitu dengan membentuk suatu pohon *Dijkstra* pada setiap pengulangan dengan menggunakan fungsi *Dijkstra-nextEdge*.
(3) Pohon *Dijkstra* yang dibentuk setelah pengulangan terakhir merupakan pohon pembangun dari graf awal.

Saran untuk penelitian selanjutnya adalah mengimplementasikan algoritma *Dijkstra* pada penentuan lintasan terpendek dalam proses evakuasi tsunami.

6. Ucapan Terima kasih

Penulis mengucapkan terima kasih kepada Dr. Mahdhivan Syafwan, Dr. Lyra Yulianti, Dr. Effendi, Dr. Admi Nazra, Dr. Haripamyu dan Dr. Des Welyyanti yang telah memberikan masukan dan saran sehingga makalah ini dapat diselesaikan dengan baik.

Daftar Pustaka

[1] Bondy, J.A. and Murty, U.S.R. 1982. *Graph Theory With Applications*. Department of Combinatorics and Optimization, University of Waterloo. North-Holland, New York.
[2] Gross. J. L., J. Yellen. *Graph Theory and Its Application Second Edition*. Chapman and Hall/CRC.
[3] West. D. B. *Introduction to Graph Theory*. Mathematics Department, University of Illinois. Prentice Hall Inc, New Jersey.