Bab 2
Tinjauan Pustaka

2.1.Gambaran Umum Penelitian Operasiaonal

2.1.1. Sejarah Singkat Perkembangan Penelitian Opersional

Pada masa Perang Dunia II, angkatan perang Inggris membentuk suatu *team* yang terdiri atas para ilmuwan untuk mempelajari persoalan-persoalan strategi dan taktik sehubungan dengan serangan-serangan yang dilancarkan musuh terhadap negaranya. Tujuan mereka adalah untuk menentukan penggunaan sumber-sumberkemiliteran yang terbatas, seperti radar dan bomber, dengan cara yang peling efektif. Karena *team* tersebur melakukan research (penelitian) terhadap operasi-operasi kemiliteran , maka munculah nama "Military Operation Research" (penelitian operasional untuk masalah kemiliteran), yang semenjak kelahirannya telah ditandai dengan digunakannya pengetahuan ilmiah dalam usaha menentukan penggunaan sumber-sumber yang terbatas.

Keberhasilan yang diperoleh angkatan perang Inggris ini kemudian mendorong angkatan perang amerika untuk melakukan aktivitas serupa, dengan membentuk *team* yang sama yang disebut *team* Operational Research. Mereka berhasil dalam memecahkan persoalan-persoalan logistiksuplai barang-barang keperluan perang, menentukan pola-pola dasar penerbangan yang lebih efisien, serta menentukan pola-pola jaringan bagi operasi alat-alat elektronik. Setelah Perang Dunia II berakhir, Operation Research yang lahir di Inggris ini kemudian berkembang pesat di Amerika karena keberhasilan yang dicapai oleh *team* Operational Research dalam bidang militer ini telah menarik perhatian orang-orang industri. Sedemikian pesat perkembangannya sehingga kini Operation Research telah digunakan hamper selurh kegiatan, baik di perguruan tinggi, konsultan, rumah sakit, perencanaan kota, maupun pada kegiatan-kegiatan bisnis.

Sebagai suatu teknik pemecahan masalah, penelitian operasional harus dipandang sebagai suatu ilmu dan seni. Aspek ilmu terletak pada penggunaan teknik-teknik

dan algoritma-algoritma matematik untuk memecahkan persoalan yang dihadapi; sedangkan sebagai seni ialah karena keberhasilan dari solusi model matematis ini sangat bergantung pada kreativitas dan kemampuan seseorang sebagai penganalisis, dalam pengambilan keputusan *(the art of balancing)*.

2.1.2. Komponen-komponen Utama Persoalan Keputusan

Munculnya persoalan-persoalan keputusan adalah karena seorang pengambil keputusan sering dihadapkan pada beberapa pilihan tindakan yang harus dilakukan. Dalam menyelesaikan persoalan yang berkaitan dengan pengambilan keputusan ini harus diindetifikasi lebih dahulu dua komponen utamanya, yaitu :

1. Objektive (tujuan),
2. Variabel-variabel. Tujuan (objective) adalah hasil akhir yang hendak dicapai dengan cara memilih suatu tindakan *yang paling tepat* untuk sistem yang dipelajari. Dalam bidang-bidang usaha biasa, tujuan diartikan sebagai "memaksimumkan profit" atau "meminimumkan ongkos yang harus dikelurkan". Akan tetapi, dalam bidang-bidang lain yang sifatnya *non-profit* (tidak mencari keuntungan), tujuan dapat berupa "pemberian kualitas pelayanan kepada para langganan".

Manakala tujuan telah didefinisikan, maka harus dilakukan pemilihan tindakan terbaik yang dapat mencapai tujuan tersebut. Dalam hal ini, kualitas pemilihan akan sangat bergantung pada apakah si pengambilan keputusan mengetahui seluruh alternatif tindakan atau tidak.

Untuk dapat menentukan tindakan-tindakanyang mungkin dilakukan itu haruslah diindetifikasikan *variabel-variabel sistem* yang dapat dikendalikan oleh pengambilan keputusan, yang keberhasilannya dalam mengindetifikasikan variabel-variabel ini pun akan sangat bergantung pada bias dan pelatihan si pengambil keputusan.

2.1.3. **Model-model Dalam Penelitian Operasional**

Model adalah gambaran ideal dari suatu situasi (dunia) nyata sehingga sifatnya yang kompleks dapat disederhanakan. Ada beberapa jenis model biasa digunakan, diantaranya ialah :

1. Model-model ikonis / fisik

   Yaitu penggambaran fisik dari suatu sistem, baik dalam bentuk yang ideal maupun dalam skala yang berbeda.

   Contoh : foto, *blueprint*, peta, globe.

2. Model-model analog / diagramatis

   Model-model ini dapat menggambarkan situasi-situasi yang dinamis dan lebih baik digunakan dari pada model-model ikonis karena sifatnyayang dapat dijadikan analogi bagi karakteristik sesuatu yang sedang dipelajari.

   Contoh : kurva distribusi frekuensi pada statistik, kurva *supply demand*, *flow chart*.

3. Model-model simbolis / matematis

   yaitu penggambaran dunia nyata melalui simbol-simbol matematis.

   Pada awalnya model matematis/simbolis ini berupa model-model abstrak yang dibentuk didalam pikiran seorang, kemudian disusun menjadi model-model simbolis, seperti gambar, symbol, atau rumus matematis. Model matematis yang paling banyak digunakan didalam penelitian operasional adalah model matematis yang berupa persamaan atau ketidaksamaan.

4. Model-model simulasi

   yaitu model-model yang meniru tingkah laku system dengan mempelajari interaksi komponen-komponennya. Karena tidak memerlukan fungsi-fungsi matematis secara eksplisit untuk merelasikan variable-variabel sistem, maka model-model simulasi ini dapat diselesaikan secara matematis. Akan tetapi, model-model ini tidak dapat memberikan solusi yang benar-benar optimum. Yang dapat diperoleh ialah jawaban yang suboptimum, yaitu jawaban optimum dari alternatif-alternatif yang dites.

5. Model-model heuristik

Kadang-kadang formulasi matematis bersifat sangat kompleks untuk dapat memberikan suatu solusi yang pasti. Atau mungkin solusi optimum yang diperoleh, tetapi memerlukan proses perhitungan yang sangat panjang dan tidak praktis . untuk mengatasi kasus seperti ini dapat digunakan *metode heuristic*, yaitu suatu metode pencarian yang didasarkan atas intuisi atau aturan-aturan empiris untuk memperoleh solusi yng lebih baik dari pada solusi yang telah dicapai sebelumnya.

Dalam penelitian operasional, model yang paling banyak digunakan adalah model matematis/simbolis. Disamping itu, digunakan juga model-model simulasi heuristik.

2.1.4. **Metodologi Penelitian Operasional**

Jika penelitian operasional akan digunakan untuk memecahkan suatu persoalan disuatu organisasi, maka harus dilakukan lima langkah sebagai berikut :

Langkah 1 : Memformlasikan persoalan

Didefinisikan persoalan lengkap dengan spesifikasi tujuan organisasi dan bagian-bagian organisasi atau system yang bersangkutan. Hal ini mutlak harus dipelajari sebelum persoalannya dapat dipecahkan.

Langkah 2 : Mengobservasikan sistem

Kumpulan data untuk mengestimasikan besaran parameter yang berpengaruh terhadap persoalan yang dihadapi. Estimasi ini digunakan untuk membangun dan mengevaluasi model matematis dari persoalannya.

Langkah 3 : Memformulasikan model matematis dari persoalan yang dihadapi

Dalam memformulasikan persoalan ini biasanya digunaksn model analitik, yaitu model matematis yang menghasilkan persamaan. Jika pada suatu situasi yang sangat rumit tidak diperoleh model analitik, maka perlu dikembangkan suatu model simulasi.

Langkah 4 : Mengevaluasi model dan menggunakannya untuk prediksi.
Pada langkah ini, tentukan apakah model matematis yang dibangun pada langkah 3 telah menggambarkan keadaan nyata secara akurat. Jika belum, buatlah model yang baru.

Langkah 5 : Mengimplementasikan hasil studi
Pada langkah ini kita harus menterjemahkan hasil studi atau hasil perhitungan ke dalam bahsa sehari-hari yang mudah dimengerti.

2.1.5. **Paket Programa Win QSB+**

Pada tahun-tahun terakhir ini teknologi Komputer berkembang dnegan cepat sekali, baik yang berkaitan dnegan perangkat keras *(hardware)* maupun perangkat lunak *(software)*.

Dalam hal perangkat lunak, saat ini telah banyak tersedia paket programa yang dapat digunakan untuk memecahkan masalah atau persoalan dalam berbagai bidang, termasuk persoalan-persoalan Operation Research (OR).

Ada beberapa paket programa yang dapat dipakai untuk memecahkan persoalan OR ini, antara lain : LP, LPV2, Simplex, QPTO, LP, LPROG, QSB, QSB+ dan Strom.

Paket programa QSB+ *(Quantitative System For Business Plus)* adalah suatu sistem yang menunjang proses mengambil keputusan *(decision support system)* yang sangat mudah untuk digunakan karena sifatnya yang interaktif. Penggunaannyapaket programa QSB+ ini memberikan beberapa keuntungan, antara lain :

1. Membantu dosen atau instruktur dalam menerangkan algoritma pemecahan masalah persoalan OR.
2. Membantu mahasiswa dalam mempelajari OR dengan cara yang lebih menarik dan menyenangkan.

3. Membantu praktisi dalam proses pengambilan keputusan.
4. Mudah dipergunakan, baik pada *personal computer* maupun *main frame.*
5. QSB+ dirancang sedemikian rupa sehingga dapat digunakan baik oleh orang yang tidak mempunyai pengalaman dalam memecahkan persoalan bisnis secara kuantitatif dengan *personal computer* maupun oleh orang yang mengenal computer dengan baik, tetapi tidak mampu membuat program computer.
6. Informasi dan pesan yang ditampilkan sangat mudah dimengerti.
7. Dapat memperlihatkan baik solusi akhir dari persoalan maupun langkah-langkah secara rinci dari proses pemecahan persoalan.
8. Menggunakan sistem menu sehingga pemakai dapat mengenal *option* yang tersedia untuk memecahkan persoalan.
9. Sistem menu memungkinkan pemakai untuk memasukan persoalan baru, membaca persoalan yang telah ada, memodifikasi persoalan yang telah ada, atau memecahkan persoalan yang ada.
10. Memungkinkan pemakai untuk memasukan data melalui *keyboard* atau membaca data dari disket jika data telah disimpan pada disket tersebut.
11. Format dirancang sedemikian rupa sehingga dapat disesuaikan *(compatible)* dengan hamper semua format pada buku-buku teks yang konvensional sehingga pemakai yang telah mempelajari konsep dasar OR akan dapat menggunakan QSB+ dengan mudah.
12. setiap programa mempunyai kemampuan untuk memodifikasi persoalan yang telah ada.

Tofik OR yang tersedia dalam paket programa QSB+ adalah sebagai berikut :

Tabel 2.1. OR dalam QSB+

No.	Topik	Kemampuan

1	*Linear Programing*	Menggunakan metode *revised simplex*, berisi antara lain : a. Pemasukan data dengan format baku atau format bebas. b. File data sesuai dengan programa QSB+ *(compact)* atau sesuai dengan IBM (MPS). c. Metode grafik untuk persoalan dengan 2 variabel. d. Metode *upper bounding* e. *Reinversion* untuk mengurangi kesalahan pembulatan. f. *Scaling* data untuk mengurangi kesalahan pembulatan. g. Penampilan table simpleks secara merinci. h. Penampilan analisis sensitivias.
2.	*Integer Linear Programing*	Menggunakan metode *branch and bound*, berisi antara lain : a. Pemasukan data dengan format baku dan format bebas. b. *File* data: *compact* atau MPS. c. *Bound* terbaru dan terbaik dari cabang terpilih. d. Metode *upper bounding*. e. *Reinversion* untuk mengurangi kesalahan pembulatan. f. *Scaling* data untuk mengurangi kesalahan pembulatan. g. Penampilan *nodes* cabang secara rinci.

3.	*Transportation dan transshipment*	Menggunakan metode *modified distribution*, berisi antara lain : a. format data masukan : bebas atau baku. b. Tujuan maksimasi atau minimasi. c. Ada 8 metode untuk menentukan solusi awal. d. *Fractional supplies* atau *demand*. e. Penampilan table-tabel rinci.
4.	*Assignment dan traveling salesman problem*	Menggunakan metode Hungarian untuk persoalan *assignment* dan metode *branch and bound* untuk masalah traveling salesman, berisi antara lain : a. format data masukan : bebas atau baku. b. Tujuan maksimasi atau minimasi. c. Penampilan table yang rinci d. Penampilan nodes cabang.
5.	Model Jaringan	Memecahkan persoalan rute terpendek, minimal spanning tree, aliran maksimal, berisi antara lain : a. format data masukan : bebas atau baku. b. Tujuan maksimasi atau minimasi. c. Penampilan langkah pemecahan persoalan secara rinci.
6.	CPM	Memecahkan lintasan kritis dan menampilkan analisis pemendekan waktu proyek *(crashing)*, berisi antara lain : a. Format table data masukan. b. Lintasan kritis ganda (multiple). c. Jaringan berdasarkan *node* atau *arc* (AON atau AOA). d. Urutan nomer *nodes* bebas. e. Titk awal dan titik akhir ganda. f. Metode simpleks untuk jadwal optimal dengan pemendekan *(crashing)*.

7.	PERT	Menampilkan PERT sampai dengan 1500 aktivitas dan analisis probabilitas, berisi antara lain : a. format table data masukan. b. Lintasan kritis ganda. c. Penampilan jaringan berdasarkan node atau *arc* (AON atau AOA). d. Urutan nomor *nodes* bebas. e. Titik awal dan titik akhir ganda.
8.	*Dynamic Programing*	Memecahkan persoalan *stagecoach*, *knapsack*, *production and inventory control* sampai 100 *stage*, berisi antara lain : a. format table data masukan. b. Penampilan pemecahan persoalan pada tiap langkah.
9.	*Inventory*	Memecahkan EOQ, EOQ dengan *discount*, dan persoalan *newsboys*, berisi antara lain : a. penampilan kurva ongkos *inventory*. b. Analisis potongan harga, baik secara total maupun *incremental*. c. Distribusi normal atau nilai diskrit untuk persoalan *newsboy stochastic.*
10.	Teori Antrian	Menganalisis persoalan antrian seperti model M/M/1, M/G/1, D/M/1, M/M/s, M/M/s/N, M/M/g/K, M/M/s/N/K, M/G/infinite, M (b)/ M/1, dan M/M/s/s, berisi antara lain : a. *Pressure service and discouraged arrival* untuk sistem M/M. b. Penampilan data umum. c. Penampilan sampai 200 probabilitas.

11.	Simulasi Sistem Antrian	Simulasi sistem antrian satu *stage* sampai dengan 20 pelayan dan 20 antrian, berisi antara lain : a. Enam macam distribusi untuk kedatangan, pelayanan. b. Spesifikasi untuk waktu simulasi, pengumpulan data waktu mulai, jumlah observasi dan bilangan random. c. Penampilan grafik untuk setiap kejadian diskrit. d. Penampilan data umum.
12.	Teori Keputusan dan Probabilitas	Memecahkan analisis variansi, rata-rata, analisis Bayesian, analisis *pay off*, pohon keputusan, berisi antara lain : a. format table data masukan. b. Penampilan langkah-langkah pemecahan. c. Ada enam criteria untuk analisis *pay off*.
13.	Proses Markov	Memecahkan masalah Markov sampai 50 *state*, berisi antara lain : a. format table data masukan. b. Penampilan tiap tahap proses memecahkan persoalan.

14.	*Time Series Forecasting*	Menampilkan peramalan *time series* dengan menggunakan berbagai metode, berisi antara lain : a. metode: *simple average, weighted moving average, moving average* dengan *linear trend, single exponential smoothing, exponential smoothing* dengan *linear trend, double exponential smoothing, double exponential smoothing* dengan *linear trend, adaptive exponential smoothing, linear regression*, dan model *winter*. b. Penampilan grafik dan hasil ramalan. c. Analisis rinci dari model tersebut. d. Pencarian secara optimal untuk parameter model.

2.2.*Minimum Spanning Tree* (Rentang Pohon Minimum)

Persoalan ini merupakan variasi dari persoalan rute terpendek yang perbeadannya terletak pada lintasan yang dicari. Pada rute terpendek, kita mencari lintasan/ rute dari sumber ketujuan yang memberikan total jarak minimum, sedangkan pada persoalan rentang pohon ini yang dipersoalkan ialah menentukan busur-busur yang menghubungkan *nodes* yang ada pada jaringan, sehingga diperoleh panjang busur total yang minimum.

Cukup banyak contoh praktis dari persoalan ini, diantaranya adalah :

a. Perencanaan jaringan transportasi

   Dalam hal ini *nodes*-nya bisa berupa terminal, sedangkan busur-busurnya dapat berupa jalan raya. Persoalannya ialah menentukan pola transportasi yang dapat melayani seluruh terminal dengan jarak yang minimum.

b. Perencanaan jaringan komunikasi berskala besar.

c. Perencanaan jaringan distribusi.

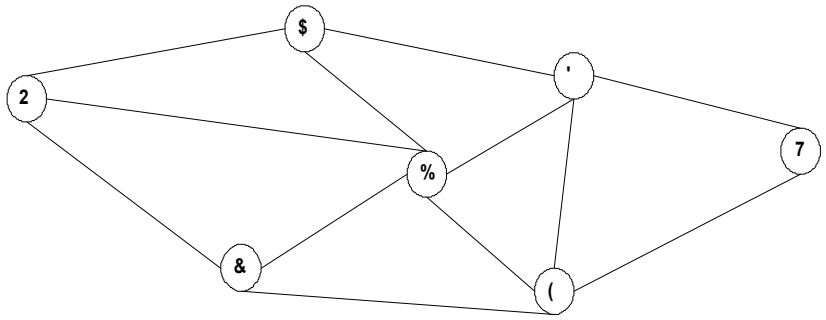
Persoalan *spanning tree* ini dapat diselesaikan dengan cara yang sangat mudah yaitu sebagai berikut :

1. pilihlah secara sembarang salah satu node, kemudian hubungkan node tersebut dengan *node* lain yang terdekat.
2. Tentukan node lain yang belum dihubungkan, yang jaraknya paling dekat dengan *node* yang sudah dihubungkan pada langkah sebelumnya kemudian hubungkan node tersebut.
3. Ulangi langkah 2 hingga seluruh *node* terhubungi.

Contoh soal :

"Alam Hijau" adalah suatu kawasan hutan lindung yang telah ditata sedemikian rupa sehingga dapat digunakan untuk hiking oleh sejumlah pecinta alam, tetapi disana pun tersedia sejenis stasiun/tram yang dikelola oleh petugas setempat. Jika dikawasan tersebut akan dipasang telepon untuk memperlancar komunikasi antar stasiun, maka dimanakah (pada jalur manakah) kabel telepon harus diletakan sehingga panjang total kabel yang akan dipasang minimum?

Penyelesaian persoalan ini akan lebih mudah jika menggunakan gambar, terlihat pada contoh gambar 2.1. dibawah ini :



Gambar 2.1. Contoh persolan *minimum spanning tree*

Dengan demikian, kabel-kabel telepon ini harus dipasang pada jalan-jalan yang menghubungkan stasiun-stasiun O dengan A, A dengan B, B dengan C, B dengan E, E dengan D, dan D dengan T, dengan demikian kabel sepanjang total 14 satuan panjang.

2.3.Transportasi

Dalam suatu perusahaan, transportasi merupakan suatu masalah yang cukup besar, terutama dikota-kota besar yang mempunyai jaringan jalan yang cukup kompleks. Alternatif-alternatif pun banyak dilakukan untuk menentukan transportasi secara efektif dan efisien. Sehingga banyak model-model matematika baru yang dikembangkan untuk menganalisis masalah-masalah yang berkaitan dengan transportasi dan merencanakan kebutuhan pada kurun waktu yang akan datang.

Masalah-masalah yang timbul dalam transportasi mengundang banyak perusahaan yang bergerak dalam bidang jasa, seperti jasa pengantaran koran, jasa telekomunikasi, jasa pos dan giro, biro perjalanan, dan lain-lain, mencari jalan keluarnya. Semua perusahaan-perusahaan tadi banyak menggunakan metode-metode transportasi.

Masalah umum dalam transportasi adalah perencanaan rute untuk kendaraan atau orang dalam melakukan perjalanan dari tempat asal ketempat tujuan. Beberapa contoh kasus penentuan rute yang harus direncanakan dalam suatu wilayah atau daerah khusus dari sebuah kota, misalnya rute distribusi surat kabar, rute kendaraan pengangkut sampah, rute pengambilan surat dari box-box surat, rute pengambilan koin telepon umum, dan lain sebagainya.

Masalah transportasi banyak digunakan model analisa jaringan yang bertujuan meminimalkan ongkos-ongkos transportasi. Penggunaan dari model tersebut telah berkembang pesat, seiring dengan berkembangnya disiplin ilmu penelitian operasional. Analisa jaringan telah menjadi alat yang cukup ideal dalam pemodelan matematis sebagai bidang ilmiah dan teknik karena mempunyai sifat yang umum dan sederhana juga bisa digunakan perhitungan secara terkomputerisasi.

Metode transportasi merupakan suatu metode yang digunakan untuk mengatur distribusi dari sumber-sumber yang menyediakan produk yang sama, ke tempat-tempat yang membutuhkan secara optimal. Alokasi produk ini harus diatur

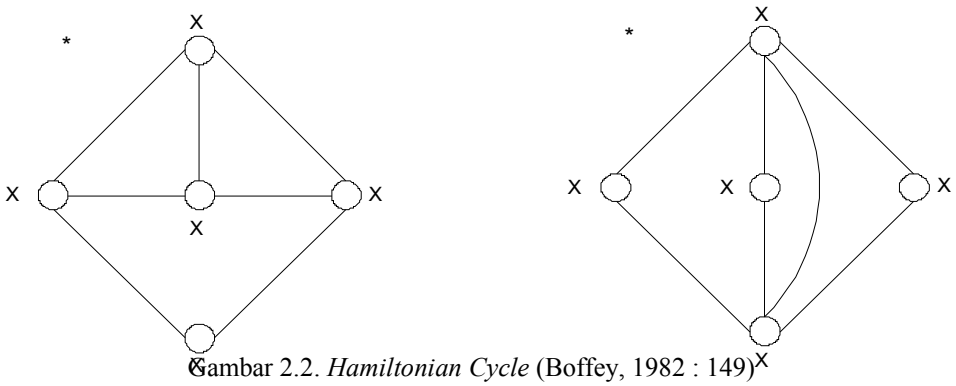
sedemikian rupa, karena terdapat perbedaan biaya-biaya alokasi dari satu sumber ke tempat-tempat tujuan yang berbeda-beda, dan dari beberapa sumber kesuatu tempat tujuan yang juga berbeda-beda. Disamping itu, metode transportasi juga dapat digunakan untuk memecahkan masalah-masalah dunia usaha (bisnis) lainnya, seperti masalah-masalahyang meliputi periklanan, pembelanjaan modal *(capital financing)* dan alokasi dana untuk investasi, analisis lokasi, keseimbangan lini perakitan dan perencanaan serta *scheduling* (penjadwalan) produksi.

"Sistem transportasi" ini dapat diwakili dengan sebuah grafik G yang ujung-ujungnya menandakan sebuah kota, dan kedua ujung-ujung tersebut disatukan oleh sebuah jalur jika dan hanya jika sebuah jalan raya menyatukan kedua buah kota yang bersangkutan dan tidak melalui kota lainnya. Solusi dari permasalahannya tergantung apakah G memiliki sebuah alur yang mencangkup setiap ujung-ujung dari G. sebuah konsep yang penting dikemukakan untuk permasalahan ini. Grafik G dinamakan sebuah grafik *Hamiltonian* jika sebuah alur terdapat pada G dan mengandung semua komponen dari G dinamakan dengan *hamiltonian cycle.* Oleh karena itu, sebuah grafik *hamiltion* adalah sebuah grafik yang mengandung sebuah *hamiltion cycle.*

Boffey (1982 : 149) menjelaskan *" A Hamiliton circuit (cycle) in a direct (underect) graph is a circuit (cycle) which visit each vertex once and once only. (Note that, in the context of traveling salesmen problems, Hamilton circuit (cycle) are often called tours.)"*

Pada gambar 2.1. berikut akan dijelaskan perbedaan grafik *Hamiltonian.* Grafik G_1 pada gambar 2.1. adalah sebuah *Hamiltonian*, sementara G_2 bukanlah sebuah *Hamiltonian*. Grafik G_1 adalah sebuah *Hamiltonian* karena terdapat sebuah *Hamiltonian cycle* ; sebagai contoh, u_1, u_2, u_5, u_4, u_3, u_1 adalah sebuah alur *Hamiltonian*. Untuk menjelaskan bahwa G_2 bukanlah sebuah *Hamiltonian*, terdapat diberikan bukti yang kontradiktif. Umpamakan pada grafik G_1 jalur dapat melewati tiap node hanya dengan mengunjunginya satu kali dan kembali ketitik

awalnya yaitu u_1, sedangkan pada grafik G_2 jika jalur ingin melewati node v_3 jalur tersebut harus melewati v_1 dan v_5 sebanyak dua kali.



Gambar 2.2. *Hamiltonian Cycle* (Boffey, 1982 : 149)

Salah satu dari masalah yang paling umum dalam sistem jasa perkotaan yaitu pada hal pembuatan rute / jalur / lintasan untuk kendaraan ataupun orang. Pada beberapa contoh misalnya, rute-rute ini harus dibuat sedemikian rupa sehingga dapat dilewati / digunakan secara maksimal pada jalan-jalan diperumahan atau pada bagian-bagian tertentu pada sebuah kota atau seringkali pada setiap sudut kota. Contoh lainnya, yaitu tujuan dibuatnya jalur / rute dapat jadi ntuk mengunjungi serangkaian poin (titik) geografis dalam sebuah kota dengan acuan untuk menyediakan beberapa jasa atau untuk mengantarkan atau untuk mengambil barang-barang..

Contoh dari tipe pertama atas permasalahan rute yang muncul termasuk didalamnya yaitu membersihkan dan menyapu jalanan, pekerjaan mengeruk salju setelah turun hujan / badai salju, pengantar surat kerumah-rumah, dan pengambilan tunggakan dari rumah-rumah. Dilain pihak, perjalanan sehari-hari dari bus sekolah, pendistribusian koran-koran pada penjualan dijalan atau kios-kios, inspeksi rutin pada pengambilan koin dari tempat-tempat telepon umum, dan pengantaran paket-paket surat untuk alamat-alamat tertentu semuanya adalah ilustrasi tipe *point-visiting* dari permasalahan rute.

Untuk alasan yang jelas, kedua buah tipe masalah tadi diatas secara berturut-turut dapat disebut dengan *edge-covering* dan *node covering problems.* Versi tertentu dari masaah tadi (selama beberapa tahun ini) telah banyak pembahasan dalam literatur matematika dan *operation research* (riset operasi). Sebagai contoh, *"traveling salesman problem"* yang paling terkenal dan paling tepat sasaran (dalam hal penjelasan masalah), merupakan versi dari masalah kelas *node-covering*, telah menjadi subjek pustaka dari ratusan laporan sains dan paper.

2.4. ***Traveling Salesman Problem*** **(TSP)**

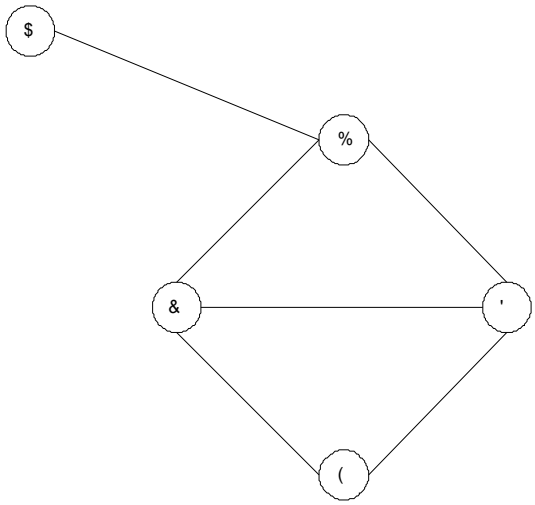
Ketika pengamatan, pengumpulan, dan kunjungan harus dilakukan ke (atau dari) sejumlah atau beberapa poin (titik) khusus (dan seringkali tersebar luas), maka masalah jalur / rute yang harus dipecahkan menjadi kasus *node-covering*. Titik permintaan (atau penyedian) kemudian dapat digambarkan sebagai node dalam *network model* (model jaringan) dari alur transportasi kota dan pertanyaannya adalah bagaimana caranya untuk mengunjungi semua node tersebut sehingga dapat mencapai tujuan yang dicari.

Sebuah permasalahan yang telah menarik banyak perhatian dan sampai sekarangpun masih demikian, adalah permasalahan *traveling salesman problem* (TSP) dimana seorang selesman memulai dari kota T_1 mengunjungi n-1 kota-kota lainnya T_2, T_3,…,T_n, dimana tiap kota hanya dikunjungi satu kali, dan kemudian kembali ke T_1. Banyak dari ketertarikan pada masalah ini muncul dari tantangan yang ada oleh fakta bahwa permasalahnnya mudah untuk ditetapkan dan dimengerti tetapi sangat sulit untuk dilakukan perhitungannya.

Yang paling mendasar dan paling dikenal dari semua kasus *node-covering* adalah *traveling salesman problem* (TSP) yang memiliki makna sebagai berikut :

> *Find the manimum distance route that begins at a given node of a network, visit all the member of a specified set of node on the network at least once, and return eventually to the initial node* (Larson, 1989 : 396).

Peryataan dari TSP diatas ini lebih bersifat umum dari pada pernyataan yang biasanya sering digunakan. Dimana belakangan ini kalimat tersebut dirubah sehingga menjelaskan bahwa setiap *node harus dikunjungi persis satu kali,* selanjutnya keterangan ini mengasumsikan bahwa ada kemungkinan untuk mendapatkan suatu jalur dengan criteria tersebut. Akan tetapi sebagai contoh, pada gambar 2.3. berikut ini bukanlah kasus jaringan yang dibicarakan diatas, dimana untuk mengunjungi *node A*, seseorang harus melewati *node B* duakali.



Gambar 2.3. Sebuah jaringan dimana seseorang *salesman* harus melewati sebuah node (node B) dua kali. (Larson, 1989:397)

Disini akan mengkhususkan untuk focus pada pemecahan TSP dalam konteks berikut : n-1 poin (node / titik) harus dikunjungi oleh sebuah poin yang spesifik pula. Jaringan transportasi yang menghubungkan n poin ini (secara keseluruhan) terhubung sempurna, maka dari pada itu memungkinkan untuk pergi langsung dari titik maupun ketitik lainnya tanpa perlu melewati titik yang sama dalam kumpulan. Jarak terdekat antara semua pasangan dari titik-titik tersebut adalah sama dengan jarak hubungan langsung antara kedua titik, contoh jika *i* dan *j* adalah titik-titik dari *n* poin kemudian $d(i,j) = l(I,j)$. Hal ini menyatakan bahwa jaringan tersebut (atau matriks perjalanan) memenuhi pertidaksamaan segitiga *(triangular inequality)* $l(i,j) \leq l(i,k) + l(k,j)$ untuk ketiga titik yang mana saja *i, j,* dan *k*. Akhirnya, matriks jarak terdekat diasumsikan simetris. Kemampugunaan

dari versi TSP untuk model selanjutnya dan untuk model jaringan dari lingkungan kota adalah jelas. Sempurnanya hubungan yang terjadi dalam jaringan dan terpenuhinya pertidaksamaan segitiga menjamin bahwa jarak terdekat jalur *traveling salesman* melalu *n* poin didapat sehingga tiap titik tepat dikunjungi satu kali.

Masalah TSP merupakan kasus yang sulit untuk dipecahkan dimana hal ini dapat dilihat dengan adanya fakta bahwa dalam versi (yang telah disederhanakan) ini TSP memiliki (*n*-1)! Urutan yang berbeda untuk mengunjungi tiap poin, yaitu (*n*-1)! / 2 solusi yang berbeda untuk TSP (dikarenakan tiap jalur dapat dilalui secara dua arah). Oleh karena itu, sebuah TSP yang hanya mempunyai 10 node saja menghasilkan 1.814.400 solusi yang memungkinkan, dan TSP yang memiliki 20 node menghasilkan sekitar $1{,}2 \times 10^{18}$.

Traveling Salesman Problem (TSP) dikenal sebgai salah satu persoalan yang sulit. Kebutuhan untuk memecahkan persoalan TSP dalam skala besar makin lama makin menaik, misalnya untuk memecahkan masalah-masalah yang berkaitan dengan rute kendaraan, rangkaian computer, penjadualan job shop dan sebagainya.

Traveling Salesman Problem (TSP) merupakan salah satu persoalan optimasi yang mempunyai diskrit yang mempunyai bidang aplikasi cukup luas dalam masalah-masalah managerial yang meliputi masalah penjadualan rute kendaraan (*vechicle routing*), rangkaian computer (*computer wiring*), penjadualan kerja (*job-shop scheduling*) dan sebagainya.
Traveling Salesman Problem (TSP) merupakan alat untuk membuat keputusan, dan persoalan yang dihadapi adalah menemukan solusi yang terbaik dari himpunan lintasan yang berhingga.

Traveling Salesman Problem (TSP) merupakan satu dari beberapa persoalan yang sulit. TSP bertujuan menemukan suatu perjalanan mengunjungi beberapa lokasi dengan tujuan meminimasikan ongkos (atau jarak maupun waktu) perjalanan.

Persoalan ini menarik perhatian, khususnya karena persoalan dengan jumlah lokasi yang kecil tampaknya yang dapat dipecahkan dalam beberapa kasus optimal pada waktu sekarang. Beberapa metode seperti Branch and Bound telah diperkenalkan dalam beberapa literature, namun metode-metode tersebut relatief mempunyai effisiensi yang rendah. Beberapa usaha lainnya adalah dengan solusi heuristik.

2.5.**Metode Heuristik Untuk TSP**

Heuristik adalah suatu metode yang tidak dapat menjamin untuk menghasilkan solusi yang optimal, tetapi dimana diharapkan menghasilkan solusi yang cukup baik. Untuk permasalahan TSP, terdapat dua macam tipe heuristik yang berbeda. Yang pertama, mencoba untuk membangun sebuah jalur perjalanan yang baik dari nol. Yang kedua, mencoba untuk memperbaiki jalur perjalanan yang sudah ada, dengan artian perbaikan setempat. Dalam prakteknya adalah sangat sulit untuk mendapatkan metode pembangunan jalur perjalanan yang benar-benar bagus. Metode dari tipe kedualah, khususnya, sebuah algoritma yang dikembangkan oleh lin dan Kernighan [1973], yang biasanya menghasilkan solusi yang paling baik dan membentuk dasar kode computer yang paling efektif.

A. Nearest Neigbor Algorithm

Nearest Neighbor Algorithm untuk TSP dapat digambarkan sebagai berikut : Mulai dari node (titik) manapun, kunjungi node terdekat yang belum dikunjungi, kemudian kembali ke node awal ketika semua node telah dikunjungi.

Johnson, Bentley, McGeoch, dan Rothberg [1997] melaporkan bahwa dalam masalah TSPLIB, rata-rata ongkos dari perjalanan yang dihasilkan oleh *Nearest Neighbor Algorithm* adalah sekitar 1,26 kali dari ongkos perjalanan yang optimal. Meskipun kemudian, untuk beberapa aplikasi, *Nearest Neighbor Algorithm* boleh jadi metode yang efektif. Metode tersebut mudah untuk digunakan, berjalan cepat, dan biasanya menghasilkan perjalanan dengan kualitas yang cukup layak. Bagaimana pun juga, perlu dicatat, bahwa perkiraan “1,26 kali optimal” adalah

pengawasan yang berdasarkan pengalaman, bukan jaminan hasilnya. Memang benar, adalah mudah untuk membuat permasalahan yang hanya memiliki empat node dimana metode *Nearest Neighbor Algorithm* dapat menghasilkan sebuah jalur ongkos yang dapat berubah-ubah setiap waktu sama dengan hasil optimumnya.

B. Insertion Methods

Insertion Methods memberikan cara yang berbeda dalam membuat jalur perjalanan. Dimulai dengan jalur perjalanan yang menggabungkan dua buah node, kemudian menambahakan node-node yang tersisa satu-persatu, dengan sedemikian rupa sehingga ongkos dari jalur tersebut meningkat dengan jumlah yang minimum. Terdapat beberapa variasi, tergantung dari kedua node awal yang dipilih, dan yang paling penting, node mana yang harus dipih untuk dimasukan tiap langkahnya.

Dalam prakteknya, biasanya metode terbaik dari *Insertion Methods* adalah *Farthest Insertion*. Dalam kasus ini, mulai dari titik awal jalur perjalanan melewati dua node dimana salah satu ujungnya adalah tepi yang mempunyai ongkos tinggi. Untuk setiap node *v* yang belum dimasukan, harus menghitung ongkos termurah antara *v* dan node lainnya dalam jalur yang telah dibuat sejauh itu. Kemudian memilih node berikutnya untuk dimasukan dimana yang satu ini memiliki ongkos maksimum.

Padamulanya, hal ini berlawanan dengan intuisi / naluri. Tetapi, dalam prakteknya, seringkali berjalan dengan baik. Hal ini kemungkinan disebabkan oleh bentuk kasar dari jalur akhir langkah, hanya sedikit perubahan yang kecil dilakukan.

Nearest Insertion adalah metode heuristik, dimana pada setiap langkahnya, memilih node berikutnya untuk dimasukan adalah node dengan ongkos paling kecil dari node lainnya.

Jenis lainnya dari *insertion method* adalah *Cheapest Insertion*. Dalam kasus ini, node berikutnya untuk dimasukan adalah yang dapat menaikan ongkos jalur paling kecil.

Biasanya solusi yang dihasilkan oleh *Nearest Insertion* dan *Cheapest Insertion* adalah lebih jelek dari yang dihasilkan oleh *Farthest Insertion.* Dalam permasalahan TSPLIB, Johnson, Bentley, McGeoch, dan Rothberg [1997] melaporkan bahwa, pada rata-ratanya, *Farthest Insertion* menemukan jalur yang menghasilkan yang optimal.

C. Christofides Heuristic

Telah digambarkan satu lagi metode pembuatan jalur, seperti yang diterangkan oleh Christofides [1976]. Metode ini mempunyai batasan yang paling baik untuk kasus yang terburuk dari pada metode lainnya yang dikenal. Metode ini selalu menghasilkan sebuah solusi yang kebanyakan 3/2 kali dari onkos optimum (dengan mengasumsikan bahwa grafiknya lengkap dan ongkos-ongkosnya tidak ada yang negatif dan memenuhi ketidaksamaan segitiga).

Algoritmanya dimulai dengan mencari sebuah *minimum-cost spanning tree*, T, dari G yang digunakan, sebuah contoh, *Kruskal's Algorithm*. Ujung-ujung dalam T adalah digunakan untuk mencari jalur yang baik.

Dimisalkan W menjadi sekumpulan dari node yang mempunyai derajat ganjil dalam T, dan mencari pasangan yang sesuai M dari G[W], subgrafik dari G yaitu W, dimana memiliki ongkos kecil yaitu c.

Sekarang misalkan J terdiri dari E(T) U M, dimana, jika beberapa ujung itu ada pada T ataupun M, lalu ambil dua buah salinan dari ujung tersebut. Kemudian J menjadi sekumpulan ujung dari sebuah grafik yang tersambung dengan sekumpulan node V yang dimana tiap nodenya mempunyai derajat genap. Jika

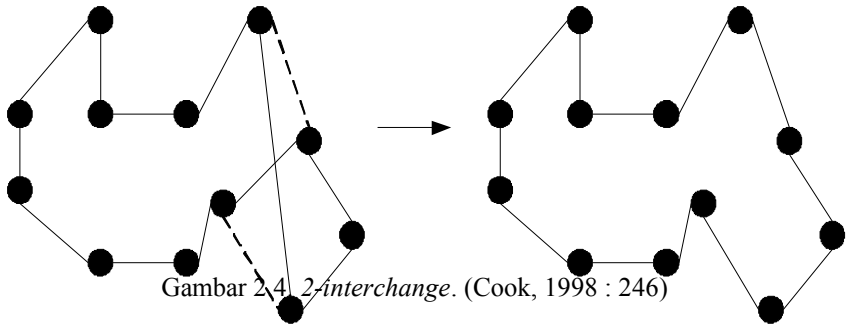
semua node mempunyai derajat 2, kemudian J adalah sekumpulan ujungnya dari jalur yang telah ditentukan. Jika sebelumnya, maka misalkan *v* sebgai node dengan setidaknya derajat 4 dalam (V,J). kemudian ada juga ujung *uv* dan *vw* dimana jika dihilangkan ujung-ujungnya ini dari J dan menambahkan ujungnya *uw* pada J, maka sub grafik pun masih terhubung. Lebih lagi, subgrafik yang baru memiliki derajat genap pada tiap nodenya. (hal ini dikarenakan *subgraph* yang dibuat oleh J mempunyai jalur *Euler*, maka pilih *uv* dan *vw* menjadi ujung yang berurutan dalam jalur). Membuat "jalan pintas" ini dan mengulangi proses ini sampai semua node bersatu dengan kedua ujungnya dari J.

Teorema / dall dari Euler adalah *"A connected graph G has an Euler tour if and only if every node of G has even degree"* Cook (1998 : 166).

Dalam tes yang dilakukan oleh Johnson, Bentley, McGeoch, dan Rothberg [1997] pada permasalahan TSPLIB, Christofides Heuristic menghasilkan jalur yang sebesar 1,14 kali dari yang optimum. Juga membuat penemuan yang menarik yaitu jika pada tiap langkah memilih jalan pintas yang terbaik untuk node yang diberikan, hasil dari algoritmanya membaik menjadi 1,09 kali dari yang optimum.

D. Tour Improvement Methods : 2-opt dan 3-opt

Terdapat beberapa metode standar dalam usaha untuk memperbaiki jalur T yang telah ada. Yang paling sederhana disebut dengan *2-opt*. Metode ini dimulai dengan mengingat tiap node yang tidak berdekatan tapi berpasangan pada ujung-ujung di T secara berurutan. Jika ujung-ujung ini dihapus, maka T akan terpecah menjadi dua buah jalan T_1 dan T_2. Terdapat sebuah cara yang unik dimana kedua jalan ini dapat disambungkan kembali untuk membentuk jalur yang baru T. Jika c (T') < c(T), maka gantikan T dengan T' dan seterusnya. Proses ini disebut dengan *2-interchange*. Dapat dilihat pada gambar dibawah ini. Jika $c(T') \geq c(T)$ untuk setiap pilihan node yang tidak berdekatan tapi berpasangan pada ujung-ujungnya, maka T adalah *2-optimal* dan segera hentikan.



Gambar 2.4. *2-interchange*. (Cook, 1998 : 246)

Algoritma *2-opt* dapat diumumkan secara ilmiah menjadi algoritma *k-opt*, dimana harus mempertimbangkan semua sub kumpulan dari sekumpulan node dalam jalur sebesar k, atau seukuran k, ganti tiap sub kumpulan pada tiap urutan, kemudian perhatikan apakah jalan yang dihasilkan dapat disatukan kembali untuk membuat jalur dengan ongkos yang lebih kecil. Masalahnya adalah jumlah sub kumpulan bertambah seiring dengan k, dan kemudian sampai pada titik pengurangan kembali. Untuk alasan inilah , *k-opr* untuk k > 3 jarang digunakan.

Johnson, Bentley, McGeoch, dan Rothberg [1997] melaporkan bahwa dalam masalah TSPLIB, *2opt* menghasilkan jalur sekitar 1,06 kali dari yang optimal dan *3-opt* sekitar 1,04 kali dari yang optimal.

E. Tour Improvement Methods : Lin-Kernighan

Lin dan Kernighan [1973] mengembangkan sebuah metode heuristik yang bekerja sangat baik dalam prakteknya. Metode ini dasarnya adalah metode *k-opt* dengan dua buah keistimewaan baru. Yang pertama, harga dari k diperbolehkan bervariasi. Kedua, ketika sebuah perbaikan ditemukan, hal itu tidak perlu digunakan secepatnya. Melainkan pencariannya diteruskan dengan harapan akan menemukan perbakan yang lebih besar dari sebelumnya.

Perlu diperhatikan bahwa *Chained Lin-Kernighan* bukanlah sebuah algoritma yang terbatas, dikarenakan tidak sediakannya peraturan penyetopan apapun. Biasanya metode ini dibiarkan berjalan sampai terlihat suatu periode yang panjang tanpa adanya perbaikan. Akan tetapi, kadang-kadang, terdapat batas byang baik

dari ongkos solusi optimum, yang akan mengakibatkan penyetopan perhitungan lebih cepat.

Dalam permasalahn TSPLIB, Johnson, Bentley, McGeoch, dan Rothberg [1997] melaporkan bahwa *Lin-Kernighan* menghasilkan jalur sekitar 1,02 kali dari yang optimal dan *Chained Lin-Kernighan* sekitar 1,01 kali dari yang optimal.

2.6. **Memecahkan Masalah TSP**

Sekarang saatnya membahas algoritma heuristik untuk TSP yaitu *Christofides heuristic*. Algoritma ini terdiri dari tiga langkah utama, dimana masing-masing langkahnya terdiri dari aplikasi dari algoritma lain yang telah dikenal. Langkah yang keempat, yang biasanya menawarkan perbaikan lebih lanjutnya dari solusi, dapat dengan mudah ditambahkan kedalam prosedur dan akan dijelaskan secara terpisah.

Ketiga langkah dasar dari algoritma heuristik ini menganalisakan sebuah jalur yang dijamin akan kurang dari 50 persen lebih panjang dari jalur yang optimal. Meskipun hal ini kedengarannya kurang menarik, ternyata ini adalah yang terbaik dalam *"wors-case performance"* (performansi memecahkan kasus terburuk) yang didapat dari algoritma TSP manapun sejauh ini.

Anggaplah bahwa n poin yang harus dilewati oleh jalur TSP (jarak simetris, hubungan yang lengkap, pertidaksamaan segitiga), maka akan didapatkan :

Algoritma Heuristik untuk TSP

Langkah 1 : Cari *minimum spanning tree* dari n poin tersebut, sebut saja dengan *minimum spanning tree* T.

Langkah 2 : Jadikan n_0 dalam n node dari T node dengan derajat ganjil (*add-degree*) (n_0 selalu merupakan angka yang genap). Carilah jarak yang paling kecil yang saling berpasang dengan node n_0, dengan menggunakan algoritma penggabungan (*pairwise matching alghoritm*). Perlu dicatat bahwa grafik yang terdiri dari hubungan perpasangan yang telah optimal disebut dengan M dan T (H = M ∪

T). perlu dicatat pula jika ada satu atau lebih hubungan yang terkadang didalam M dan T, hal ini akan terlihat pada H.

Langkah 3 : Grafik H adalah sebuah grafik *Eulerian*, karena didalamnya tidak terdapat node yang ganjil. Gambarlah jalur Eulerian pada H (dimulai dan diakhiri pada node awal yang ada pada jalur TSP, jka node awalnya telah ditentukan sebelumnya). Jalur *Eulerian* ini (kurang lebih) adalah solusi dari TSP.

Teorema *Euler's : "A connected graph G possesses an Euler tour (Euler path) if and only if G contains exactly zero (exactly two) nodes of add degree"*(Larson, 1989:396).

2.7.**Permasalahan *Multiroute***

Sejauh ini yang telah didiskusikan hanyalah permasalahan rute tunggal dari tipe *node-covering*. Kenyataannya sering kali, harus berhadapan dengan jalur keberangkatan kendaraan yang bukan hanya satu tetapi beberapa buah, yang harus berbagi tugas dalam menyediakan jasa kepada suatu wilayah tertentu. Dalam hal ini pengambilan sampah, sebagai contoh yang konkrit, sebuah departemen kebersihan harus memikirkan jalur bagi kendaraan-kendaraannya untuk mengambil sampah tiap daerah pada sebuah kota yang telah terbagi-bagi. Dari contoh diatas maka dikenal tipe permasalahan rute lainnya yaitu *multiroute node – covering*.

Perlu diketahui juga bahwa pada permasalahannya algoritma yang tersedia untuk permasalahaan *multiroute* adalah algoritma heuristic. Secara garis besar, permasalahan *multiroute* yaitu mengkombinasikan algoritma rute tunggal yang ada pada kasus *node-covering*.

Multiroute Node-Covering

Akan sangat membantu, jika pada tahap ini menyatakan klasifikasi dari permasalahan *node-covering***.** Penjelasan dasar dari permasalahan *node covering* ada tiga yaitu :

1. Jumlah dari kendaraan (*salesman*).
2. Jumlah titik awal jalur (yang juga berarti tujuan jalur).
3. Adanya batasan terhadap sebuah barang seperti kapasitas kendaraan pribadi (*salesman*), jarak maksimum dari sebuah jalur, dan sebaginya.

Secara umum, permasalahan yang tidak memiliki batasan seperti yang disebutkan no.3 diatas dikenal dengan permasalahan (*salesman*). Jadi permasalahan klasik TSP adalah masalah dimana sebuah jalur minimum harus didesain untuk kendaraan *salesman* tunggal dengan menggunakan satu titik awal tujuan tanpa adanya batasan kapasitas atau batasan jarak jalur.

Yang dikenal dengan *multiroute-traveling salesman problem* (*m*-TSP), dengan kata lain melibatkan pembatan sejumlah rute tertentu (*m*), yang bersama-sama mengunjungi tiap poin-poin demand setidaknya satu kali dengan tujuan awal dan akhir yang sama. Dimana tujuannya tersebut untuk meminimasi jarak total yang dicangkup oleh jalur *m*.

Perlu diingat bahwa permasalahan *multiroute-traveling salesman problem* adalah permasalahan dengan tujuan awal dan akhir yang sama, dimana dapat ditunjukan dengan memformulasikan ulangnya sebagai TSP klasik dengan sedikit tambahan. Yaitu beberapa TSP klasik (tunggal) dengan tujuan awal dan akhir yang sama disatukan dalam sebuah grafik.