# STUDI TENTANG *TRAVELLING SALESMAN* DAN *VEHICLE ROUTING PROBLEM* DENGAN *TIME WINDOWS*

**I Nyoman Sutapa, I Gede Agus Widyadana**
Dosen Fakultas Teknologi Industri, Jurusan Teknik Industri, Universitas Kristen Petra

**Christine**
Alumnus Fakultas Teknologi Industri, Jurusan Teknik Industri, Universitas Kristen Petra

## ABSTRAK

Dalam artikel ini dipaparkan kajian mengenai pengembangan model *travelling salesman problem*. Ada tiga model yang dikaji yaitu *travelling salesman problem* dengan *time windows*, *vehicle routing problem*, serta *vehicle routing problem* dengan *time windows*.

**Kata-kunci**: *travelling salesman problem, vehicle routing problem, time windows.*

## *ABSTRACT*

*The article shows the study of model development of travelling salesman problem. Three models are studied, i.e. travelling salesman problem with time windows, vehicle routing problem, and vehicle routing problem with time windows.*

***Keywords:*** *travelling salesman problem, vehicle routing problem, time windows.*

## 1. PENDAHULUAN

Balas (1996) mendefinisikan *travelling salesman problem* (TSP) sebagai berikut: diberikan suatu *initial ordering* dari *n* kota, dan sebuah integer $k>0$, kemudian dicari biaya minimal dari *feasible tour*. *Feasible tour* yang dimaksud adalah *tour* dimana kota *i* didatangi sebelum kota *j*, jika $j \geq i+k$, dalam *initial ordering*. Dalam hal ini diketahui *N* titik yang disebut kota, dan $c_{ij}$ biaya perjalanan dari *i* ke *j*, untuk semua $i, j \in N$. Tujuan yang ingin dicapai adalah mencari biaya permutasi atau *tour* minimal, dari semua kota. Jika $c_{ij}=c_{ji}$, maka TSP tersebut simetris, sebaliknya disebut asimetris. TSP merupakan permasalahan mencari *Hamiltonian cycle* terpendek dalam graph (berarah atau tidak berarah).

Untuk menyelesaikan permasalahan TSP, Pesant (1998) mengembangkan sebuah model *constraint proggramming*. Parameter dan variabel yang digunakan dalam model ini adalah: $V=\{2,...,n\}$ yang mewakili kota–kota yang akan dikunjungi, dengan kota asal atau *origin-depot* dinyatakan dengan $V=1$, sedangkan untuk *destination depot* dinyatakan dengan $V=n+1$. Suatu *tour* akan merupakan *Hamiltonian path* yang berawal di *1* dan berakhir di $n+1$, yang dapat dinyatakan sebagai:

$$V^o = V \cup \{1\}$$

$$V^d = V \cup \{n+1\}$$

$$V^{o,d} = V \cup \{1, n+1\}$$

Selanjutnya, $c_{ij}$ menunjukkan biaya *travel* dari kota *i* ke kota *j*, dan bagian tengah dari model merupakan variabel $S_i$, $i=1,...,n,$ yang berkaitan dengan tiap kota (dan *origin-depot*), dan yang menunjukkan *immediate successor* dalam sebuah tour. Sehingga domainnya merupakan bilangan integer antara $2, ..., n+1$.

Lebih lanjut, sebuah *tour* yang *valid*, memberikan *distinct successor* pada tiap kota dan menghindari adanya *sub-tour*. Dan, bila $\beta_i$ menunjukkan awal dari *partial part* yang melalui *i*, sedangkan $\varepsilon_i$ menunjukkan akhirnya; pada awalnya $\beta_i=\varepsilon_i = i$.

Model *Constraint proggramming* untuk TSP dapat dirumuskan dengan:

Fungsi tujuan: $Min \quad \sum_{i \in V^o} c_i, s_i$ (1)

dengan kendala-kendala:

$S_i \neq S_j, \qquad \forall i, j \in V^o, i \neq j$ (2)

$S_i \neq i, \qquad \forall i \in V^o$ (3)

$S_i = j \Rightarrow S_{\in_j} \neq \beta_i, \ (\in_j \neq \mathrm{n}+1) \qquad \forall i \in V^o$ (4)

$S_i \in \{2,...,n+1\}, \qquad \forall i \in V^o$ (5)

Fungsi tujuannya adalah meminimalkan total biaya dari *tour* $c_i$, $s_i$ yang merupakan biaya *travel* dari *i* ke *immediate successor Si* (1). Kendala (2), dan (5) memastikan bahwa tiap kota dikunjungi hanya satu kali saja, sedangkan kendala (3), dan (4) digunakan untuk menghilangkan *sub-tour*.

Kendala (5) hanya digunakan untuk menentukan *domain*. Kendala (3) digunakan untuk menghilangkan *i* dari *domain* pada tiap–tiap $S_i$. Kendala (2) menunggu sampai $S_i$ atau $S_j$ menjadi *fixed* pada nilai *k*, kemudian kendala ini akan menghilangkan *k* dari *domain* variabel lain. Pada kendala (4), saat $S_i$ *fixed* pada *j*, *partial path* yang berakhir pada *i* akan bergabung dengan *partial path* lain yang berawal dari *j*. Karena *sub-tours* dilarang, maka akhir dari *path* baru, $\in_j$, tidak dapat diikuti oleh awalnya, $\beta_i$. Pada kasus khusus, jika $\in_j=n+1$, yang berarti bahwa *path* sudah mencapai *destination depot*, maka tidak ada lagi yang perlu dilakukan.

## 2. *TRAVELLING SALESMAN PROBLEM* DENGAN *TIME WINDOWS*

Pesant (1998) mendefinisikan *travelling salesman problem* dengan *time windows* (TSPTW) sebagai permasalahan untuk mencari biaya *tour* minimal dari sekumpulan kota, dimana tiap kota hanya dikunjungi satu kali saja. Agar *feasible*, maka *tour* tersebut harus berawal dan berakhir disuatu *depot* tertentu, dalam batas *time window* tertentu, dan tiap kota harus dikunjungi pada batas *time window* mereka masing–masing. Biaya TSPTW biasanya berhubungan dengan total jarak *travel* atau total waktunya (waktu *travel* ditambah waktu tunggu ditambah waktu pelayanan). Jadi model TSPTW merupakan pengembangan TSP, yaitu dengan tambahan kendala *time windows* untuk masing–masing kota. *Time windows* $[a_i, b_i]$ menunjukkan batas waktu pelayanan di kota *i*, dengan batas awal $a_i$ dan batas akhir $b_i$. Kedatangan sebelum $a_i$ diperbolehkan tetapi mengakibatkan adanya waktu tunggu sampai batas *time windows*, tetapi tidak diperbolehkan adanya kedatangan sesudah $b_i$.

Perumusan *constraint proggramming model* dari TSPTW sama dengan model model TSP diatas, dengan tambahan kendala:

$$a_i \le T_i \le b_i, \qquad \forall i \in V^d \tag{6}$$

$$Si = j \Rightarrow T_i + t_{ij} \le T_j \qquad \forall i \in V^o \tag{7}$$

dimana $t_{ij}$ adalah waktu *travel* antara kota *i* dan *j*, sedangkan $T_i$, adalah waktu mulainya pelayanan di kota *i* dengan $i = 1,...,n+1$, dan $T_1 = 0$.

Kendala (6) membatasi agar waktu pelayanan sesuai dengan *time windows*-nya, sedangkan kendala (7) untuk memastikan feasibilitas dari jadwal yang dibuat.

## 3. *VEHICLE ROUTING PROBLEM* (VRP)

Kallehauge dkk. (2001) mendefinisikan permasalahan *m*-TSP sebagai salah satu variasi dari TSP, dimana terdapat *m-salesman* mengunjungi sejumlah kota dan tiap kota hanya dapat dikunjungi oleh tepat satu *salesman* saja. Tiap *salesman* berawal dari suatu *depot* dan pada akhir perjalannya juga harus kembali ke *depot* tersebut.

Permasalahan *m*-TSP sering disebut sebagai *vehicle routing problem* (VRP), dimana sebuah kota diasosiasikan sebagai sebuah *demand* atau konsumen, dan tiap kendaraan yang dipakai untuk perjalanan dianggap memiliki kapasitas tertentu. Total jumlah *demand* dalam suatu *rute*, tidak boleh melebihi kapasitas dari kendaraan yang ditugasi melewati *rute* tersebut. Hal ini membuat VRP kadang juga disebut sebagai *Capacitated Vehicle Routing Problem*. Sama seperti permasalahan TSP, dalam VRP juga terdapat suatu *depot*, dimana tiap kendaraan harus berangkat dan kembali ke *depot* itu. Dalam VRP, selain bertujuan untuk meminimalkan total jarak atau total biaya *travel*, dapat juga untuk meminimalkan jumlah kendaraan yang digunakan (*m*).

Prins (2001) menggambarkan permasalahan VRP sebagai suatu *undirected network* $G=(V, E)$, dengan sebuah *node set* $V=\{0, 1, ..., n\}$, dan sebuah *edge set E*. *Node 0* adalah sebuah *depot*, dengan sejumlah kendaraan yang mempunyai kapasitas yang sama atau identik, *Q*. Tiap klien *node i>0*, memiliki suatu *demand non negatif* $q_i$, dan *tiap edge* $[i, j]$ memiliki biaya *non negatif* $c_{ij}=c_{ji}$. Permasalahan VRP bertujuan untuk menentukan suatu *set trips* kendaraan dengan total biaya minimal, dimana tiap *trip* berawal dan berakhir di *depot*, tiap klien dikunjungi tepat satu kali, total *demand* yang dibawa oleh tiap kendaraan tidak melebihi kapasitas kendaraan *Q*, dan biaya dari tiap *trip* tidak melebihi *upper limit L*, yang telah ditentukan. Pada penelitian yang dilakukan oleh Prins (2001), variabel keputusannya adalah jumlah kendaraan.

Thangiah (1995) merumuskan model *mixed-integer programing* untuk permasalahan VRP. Parameter yang digunakan antara lain: *K* sebagai nomer kendaraan, *N* sebagai nomer konsumen (*0* menunjukkan *depot*), $C_i$ menunjukkan konsumen *i*, $C_0$ menyatakan *depot*. Selanjutnya $V_k$ sebagai rute kendaraan *k*, $c_{ijk}$ adalah biaya *travel* antara konsumen *i* dan *j* untuk kendaraan *k*, $q_{ik}$ menyatakan total demand kendaraan *k* sampai konsumen *i*, dan $v_k$ adalah kapasitas maksimum kendaraan *k*.

Sedangkan, variabelnya didefinisikan sebagai:

$$y_{ik} = \begin{cases} 1 & \text{jika konsumen i dilayani oleh kendaraan k} \\ 0 & \text{jika tidak demikian} \end{cases}$$

$$x_{ijk} = \begin{cases} 1 & \text{jika kendaraan } k \text{ dari konsumen } i \text{ langsung ke konsumen } j \\ 0 & \text{jika tidak demikian} \end{cases}$$

Formulasi *mixed-integer programing* untuk VRP ini adalah:
Fungsi tujuan:

$$Min \sum_{i=0}^{\mathrm{N}} \sum_{j=0}^{N} \sum_{\mathrm{k}=1}^{K} c_{ij} x_{ijk} \tag{8}$$

denaga kendala-kendala:

$$\sum_{i=0}^{N} q_{ik} y_{ik} \le v_k \,,\ k = 1,\ldots,K \tag{9}$$

$$y_{ik} = 0 \text{ atau } 1 \ ;\ i = 1, 2, \ldots, N \,;\, k = 1, 2, \tag{10}$$

$$x_{ijk} = 0 \text{ atau } 1 \ ;\ i = 1, 2, \ldots, N \,;\, k = 1, 2, \ldots, K \tag{11}$$

$$\sum_{k=1}^{K} y_{ik} = \begin{cases} K, \ i = 0 \\ 1, \ i = 1, \ldots, N \end{cases} \tag{12}$$

$$\sum_{i=0}^{N} x_{ijk} = y_{jk} \,,\ j = 0, \ldots, N;\ k = 1, \ldots, K \tag{13}$$

$$\sum_{j=0}^{N} x_{ijk} = y_{ik} \,,\ i = 1, \ldots, N;\ k = 1, \ldots, K \tag{14}$$

Tujuan dari model ini adalah untuk meminimalkan total biaya *travel*. Kendala (9) membatasi bahwa total jumlah *demand* yang dibawa oleh kendaraan $k$ tidak boleh melebihi kapasitas dari kendaraan tersebut. Kendala (12) untuk menunjukkan bahwa tiap konsumen hanya dapat dilayani oleh satu kendaraan saja. Kendala (13), dan (14) digunakan untuk memastikan bahwa tiap konsumen dikunjungi oleh kendaraan yang sama dengan yang sudah dijadwalkan untuk konsumen tersebut.

Sedangkan, Kallehauge dkk.(2001) memodelkan VRP sebagai:

Fungsi tujuan: $\text{Min} \sum_{k=1}^{K} \sum_{i=0}^{N+1} \sum_{j=0}^{N+1} c_{ij} x_{ijk}$ (15)

dengan kendala-kendala:

$$\sum_{k=1}^{K} \sum_{j=0}^{N+1} x_{ijk} = 1; \qquad i = 1, 2, \ldots, N \tag{16}$$

$$\sum_{i=1}^{N} d_i \sum_{j=0}^{N+1} x_{ijk} \le v_k; \qquad k = 1, 2, \ldots, K \tag{17}$$

$$\sum_{j=0}^{N+1} x_{ojk} = 1; \qquad k = 1, 2, \ldots, K \tag{18}$$

$$\sum_{i=0}^{N+1} x_{ihk} - \sum_{j=0}^{N+1} x_{hjk} = 0; \qquad h = 1, 2, \ldots, N \,;\, k = 1, 2, \ldots, K \tag{19}$$

$$\sum_{i=0}^{N+1} x_{i,N+1,k} = 1 ; \quad k = 1, 2, ..., K \tag{20}$$

$$x_{ijk} \in \{0,1\} ; \qquad i = 0, 2, \ldots, N + 1 ; k = 1, 2, \ldots, K \tag{21}$$

Fungsi tujuan dari model VRP Kallehauge dkk. (2001), sama seperti fungsi tujuan dari model yang dibuat Thangiah (1995), yaitu untuk meminimalkan total biaya *travel*. Tetapi Thangiah hanya memperhitungkan biaya *travel* untuk perjalanan awal dari *depot*, kemudian mengunjungi semua konsumen yang ada saja, tanpa memperhitungkan perjalanan kembali ke *depot*, pada akhir perjalanan tersebut. Sedangkan Kallehauge dkk. (2001), memperhitungkan biaya *travel* untuk perjalanan awal dari *depot*, kemudian mengunjungi semua konsumen yang ada saja, dan juga perjalanan kembali ke *depot*, pada akhir perjalanan tersebut.

Kendala (16) mempunyai kegunaan yang sama seperti kendala (12), yaitu untuk menunjukkan bahwa tiap konsumen hanya dapat dilayani oleh satu kendaraan saja. Kendala (17) mempunyai kegunaan yang sama seperti kendala (9), yaitu untuk membatasi bahwa total jumlah *demand* yang dibawa oleh kendaraan *k*, tidak boleh melebihi kapasitas dari kendaraan tersebut. Kendala (18)–(20) digunakan untuk memastikan bahwa tiap kendaraan berangkat dari *depot* 0, dan setelah selesai melayani seorang konsumen, kendaraan tersebut akan pergi, serta pada akhirnya, kendaraan tersebut akan kembali ke *depot N+1*. Pada model yang dibuat oleh Tangiah (1995), tidak terdapat kendala yang mempunyai kegunaan seperti kendala (18)–( 20).

## 4. *VEHICLE ROUTING PROBLEM* DENGAN *TIME WINDOWS*

Menurut Kallehauge dkk. (2001), *vehicle routing problem* dengan *time windows* (VRPTW) adalah perluasan dari VRP. Jika pada VRP ditambahkan *time window* pada masing–masing konsumen, maka permasalahan tersebut menjadi VRPTW. Untuk VRPTW, selain adanya kendala kapasitas kendaraan, terdapat tambahan kendala yang mengharuskan kendaraan untuk melayani tiap konsumen pada *time frame* tertentu. Kendaraan boleh datang sebelum *time window* "*opens*", tetapi konsumen tersebut tidak dapat dilayani sampai *time window* "*opens*". Kendaraan tidak diperbolehkan untuk datang setelah *time window* "*closed*".

Tangiah (1995) mendefinisikan VRPTW sebagai permasalahan untuk menjadwalkan sekumpulan kendaraan, dengan kapasitas dan *travel time* terbatas, dari *central depot* kesekumpulan konsumen yang tersebar secara geografis, dengan *demand* diketahui, dalam *time windows* tertentu. *Time windows* adalah *two sided*, yang berarti bahwa tiap konsumen harus dilayani saat atau setelah *earliest time*, dan sebelum *latest time* dari konsumen tersebut. Jika kendaraan datang ke konsumen sebelum *earliest time* dari konsumen tersebut, maka akan menghasilkan *idle* atau waktu tunggu. Kendaraan yang datang ke konsumen setelah *latest time* adalah *tardy*. Terdapat pula waktu *service* yang diperlukan untuk melayani tiap konsumen. Biaya rute dari suatu kendaraan adalah total dari waktu *travel* (proposional dengan jarak), waktu tunggu, dan waktu *service*, yang diperlukan untuk mengunjungi sekumpulan konsumen.

Homberger dkk. (1999) mendefinisikan permasalahan VRPTW sebagai berikut: *n* konsumen akan dilayani dari sebuah *depot*, dengan sejumlah kendaraan yang memiliki kapasitas, *Q*, yang sama. Untuk tiap konsumen *i*, *i*=1, 2, ..., *n*, terdapat *demand* $q_i$, waktu *service* $s_i$, dan *service time window* $z_i$=[$e_i$, $f_i$]. *Lower bound* $e_i$ merupakan waktu paling awal untuk melakukan *service*, dan *upper bound* $f_i$, waktu paling lambat untuk melakukan *service*. *Demand* $q_i$ dari konsumen *i* harus dipenuhi dengan sekali *service* saja, dalam batas *time window* $z_i$. Sebagai tambahan, $e_0$ merupakan waktu paling awal untuk kendaraan berangkat dari *depot i, i=0*, dan $f_0$ merupakan waktu paling lambat untuk kendaraan kembali ke *depot*. Data mengenai lokasi dari *depot*, dan konsumen, jarak terpendek $d_{ij}$, serta waktu *travel* $d'_{ij}$ antara dua lokasi, diketahui. Tujuannya untuk menentukan jadwal rute yang *feasible*, yaitu pertama untuk meminimalkan jumlah kendaraan, dan kedua untuk meminimalkan total jarak *travel*. Konsumen tidak dapat dilayani diluar *time window* mereka masing–masing. Tetapi kendaraan diperbolehkan untuk datang sebelum *lower bound* dari *time window*. Apabila hal ini terjadi, maka kendaraan harus menunggu sampai batas waktu paling awal *service* tersebut dapat dilakukan.

Ketiga definisi VRPTW diatas, membahas VRPTW dengan *hard time windows*, yaitu tiap kendaraan diperbolehkan untuk sampai ke konsumen *i* sebelum waktu pelayanan paling awal konsumen tersebut ($e_i$), tetapi tidak diperbolehkan datang melewati batas waktu pelayanan paling akhir konsumen itu ($l_i$). Bila kendaraan datang ke konsumen *i* sebelum batas waktu $e_i$, maka akan dikenakan waktu tunggu sampai batas waktu $e_i$ tersebut.

Untuk memodelkan VRPTW, Tangiah memperkenalkan beberapa parameter yang perlu ditambahkan, yaitu: $R_k$ sebagai total waktu rute untuk kendaraan *k*, $t_{ij}$ untuk waktu *travel* antara konsumen *i* dan *j* (proporsional dengan jarak *Euclidean*), $t_i$ menyatakan waktu kedatangan di konsumen *i*, $f_i$ sebagai waktu *service* di konsumen *i,* $w_i$ adalah waktu tunggu sebelum melayani konsumen *i,* $w_i=max\{0\ ,(e_i–t_i\ )\}$, $e_i$ sebagai waktu paling awal untuk pelayanan di konsumen *i,* dan $l_i$ adalah waktu paling akhir untuk pelayanan di konsumen *i.*

Seperti halnya dengan VRP, fungsi tujuan VRPTW yaitu meminimalkan total biaya *travel* semua kendaraan (8). Sedangkan semua kendalanya juga sama dengan kendala VRP [(9)-(14)], tetapi perlu ditambahkan beberapa kendala lagi yang berhubungan dengan kendala *time windows*. Kendala–kendala yang perlu ditambahkan tersebut adalah:

$$\sum_{i=0}^{N}\sum_{i=0}^{N} y_{ik}\,(t_{ij} + f_i + w_i) \le R_k \ ; k = 1, ..., K \tag{22}$$

$$t_j \ge t_i + w_i + f_i + t_{ij} - M\left(1 - x_{ijk}\right) ; \quad i,j = 1,...,N;\ k\ =\ 1,..., K \tag{23}$$

$$e_i \le t_i < l_i ;\ i = 1, ..., N \tag{24}$$

$$t_i \ge 0 ;\ i = 1, ..., N \tag{25}$$

Kendala (22) digunakan untuk memastikan bahwa tiap kendaraan melayani semua konsumen yang dijadwalkan untuk kendaraan tersebut, tanpa melebihi waktu *travel* dari kendaraan tersebut. Kendala (23) untuk memastikan waktu kedatangan dari kedua konsumen adalah *compatible*. *M* merupakan bilangan riil yang sangat besar. Kendala (24) mengharuskan kendaraan untuk sampai di tiap–tiap konsumen selama batas *time window*

dari konsumen tersebut. Kendala (25) memastikan bahwa waktu kedatangan kendaraan ke tiap konsumen selalu positif.

Untuk memodelkan VRPTW, Kallehauge dkk. (2001) juga menambahkan beberapa kendala ke model permasalahan VRP sebelumnya, yaitu yang terdapat pada persamaan [(15) – (21)]. Kendala yang ditambahkan tersebut adalah :

$$t_i + t_{ij} - M(1 - x_{ijk}) \le t_j \;;\; i, j = 0, 1, ..., N+1 \;;\; k = 1, 2, ..., K \tag{26}$$

$$e_i \le t_i \le l_i \;;\; \; i = 0, 1, ..., N+1 \tag{27}$$

Parameter $t_{ij}$ yang digunakan pada model ini memiliki arti yang berbeda dari parameter $t_{ij}$ yang digunakan pada model yang dibuat oleh Tangiah. Pada model yang dibuat oleh Kallehauge dkk. (2001) ini, $t_{ij}$ berarti suatu waktu yang dimiliki oleh setiap *arc*$(i, j)$, dimana $i \neq j$. Jadi, $t_{ij}$ pada model ini memuat waktu perjalanan antara konsumen $i$ dan $j$, waktu tunggu di konsumen $i$, serta waktu *service* di konsumen $i$. Kendala (26) mirip dengan kendala (23), dan juga mempunyai kegunaan yang sama dengan kendala (23), yaitu untuk memastikan bahwa waktu kedatangan dari kedua konsumen adalah *compatible*. Perbedaaan antara kendala (23), dan (26) adalah, bahwa kendala (26) mengikut sertakan *depot*, sedangkan kendala (23) hanya untuk konsumen saja, tanpa mengikut sertakan *depot*. Sedangkan kendala (27) juga mirip dengan kendala (24), dan juga mempunyai kegunaan yang sama dengan kendala (24), yaitu untuk mengharuskan kendaraan untuk sampai di tiap–tiap konsumen selama batas *time window* dari konsumen tersebut. Perbedaaan antara kendala (24), dan (27) adalah, bahwa kendala (27) mengikut sertakan *depot*, sedangkan kendala (24) hanya untuk konsumen saja.

Untuk menyelesaikan permasalahan VRPTW ini, Braysy (2002) menyarankan suatu algoritma yang berdasarkan pada pendekatan tiga fase. Fase pertama adalah *initial solution* yang dibuat menggunakan salah satu dari dua *route construction heuristics* yang disarankan, yaitu *hybrid construction* atau *merge heuristic*. Pada fase kedua, dilakukan pengurangan jumlah rute dengan menggunakan *local search operator* yang berdasarkan *ejection chains*. Akhirnya, pada fase ketiga, *or-opt exchanges* digunakan untuk meminimalkan total jarak yang ditempuh untuk tiap rute. Urutan langkah dari algoritma tersebut adalah sebagai berikut:

(1) Membuat *initial solution* dengan *hybrid construction* atau *merge heuristic*.
(2) Mengulangi prosedur pengurangan rute, sampai tidak ada lagi rute yang dapat dihilangkan.
(3) Mengulangi langkah pertama dan kedua menggunakan semua nilai parameter dalam batas limit tertentu. Hasil solusi yang dibuat tersebut kemudian disimpan.
(4) Dari semua hasil solusi yang telah diperoleh, dicari solusi dengan jumlah rute paling sedikit, dan dimasukkan dalam set *RB*.
(5) Menata ulang urutan dalam solusi $S_i$, menurut parameter $\beta$, dan memperbaiki $S_i$ dengan menggunakan prosedur *or-opt*.
(6) Mengulang langkah kelima untuk semua $S_i$ dalam *RB* dan meng-*update* solusi terbaik yang ditemukan, $S_b$, jika dibutuhkan.
(7) Kembali ke $S_b$.

Metode yang diusulkan ini telah diujikan pada 56 *test problems of Solomon*. Hasil pengujian tersebut kemudian dibandingkan dengan hasil dari metode *local searches* dan *metaheuristics* terbaik yang telah ditemukan sebelumnya, di dalam literatur. Dari hasil perbandingan tersebut, terlihat bahwa metode yang diusulkan oleh Braysy (2002) ini

sangat efisien untuk menyelesaikan permasalahan VRPTW, menghasilkan hasil yang lebih baik dari pendekatan *local search* sebelumnya, dan selain lebih cepat, metode ini juga mampu bersaing dengan *metaheuristics* yang terbaik, dalam hal kualitas dari solusinya.

Alvarenga dkk. (2003) mengusulkan suatu pendekatan *robust heuristic* untuk permasalahan VRPTW menggunakan *genetic algorithm (GA)* yang efisien dan formulasi *mix integer programing (MIP)*. Untuk GA yang digunakan, *chromosome*-nya berupa sebuah *string of integer*. Untuk tiap kendaraan dengan minimal satu konsumen yang harus dikunjungi, dialokasikan satu *chromosome*. Sebuah individu, yang mempresentasikan sebuah solusi yang lengkap, dan seringkali berisi banyak rute, merupakan kumpulan dari *chromosome*. Untuk menentukan *initial population*, digunakan metode heuristik *push forward insertion heurist (PFIH)*. Pada tahap *selection*, dipilih sepasang *parents* untuk *crossover*. Pada penelitiannya, untuk proses *selection*, digunakan *k-way tournament selection method*. Dalam metode ini, sejumlah *k* individu dipilih secara random. Kemudian, individu yang mempunyai *fitness* paling tinggilah pemenangnya. Proses ini diulangi sampai jumlah individu yang dipilih mencukupi jumlah individu yang dibutuhkan untuk *crossover*.

Christine (2004) memecahkan pendekatan yang diusulkan Alvarenga diatas menggunakan *evolutionary algorithm* (EA) dengan *crossover* yang langkah–langkahnya adalah sebagai berikut:

(1) Pada langkah pertama, dibuat sebuah pilihan rute secara *random* untuk tiap *parent individual*.
(2) Setelah semua rute yang *feasible* dimasukkan, untuk tiap konsumen yang tertinggal, akan diuji apakah dapat disisipkan pada rute yang tidak kosong dari individu baru.
(3) Apabila masih terdapat beberapa konsumen yang tidak dapat dimasukkan ke dalam rute yang ada, maka harus dibuatkan rute baru dalam individu baru tersebut.

Untuk proses *mutation*, digunakan delapan operator yang berbeda, yaitu: (1) *Random Customer Migration, (2) Bringing the Best Customer, (3) Re-insertion using PFIH, (4) Similar Customer Exchange, (5) Exchanging Customer with Positive Gain, (6) Merging Two Routes, (7) Reinserting Customer,* dan (8) *Route Partitioning*

Algoritma yang diusulkan oleh Alvarenga (2003) tersebut telah dibuktikan sangat *robust*, dengan hasil yang selalu kurang dari 1% terhadap solusi optimum (dari beberapa metode *exact* yang terdapat dalam literatur). Sedangkan untuk permasalahan yang tidak ditemukan solusi *exact*-nya, hasil dari *hybrid* GA ini seringkali lebih baik dari hasil metode *heuristic* yang ada.

## 5. KESIMPULAN

Dalam artikel ini telah dibahas model-model pengembangan *travelling salesman problem*, yaitu *travelling salesman problem* dengan *time windows*, *vehicle routing problem*, dan *vehicle routing problem* dengan *time windows*. Beberapa model yang dibahas diatas merupakan model-model standar, dengan harapan kajian pendahuluan ini bermanfaat sebagai acuan dasar dalam pemecahan masalah terutama pada manajemen rantai pasok.

## DAFTAR PUSTAKA

Alvarenga, G.B., G.R. Mateus, dan G. de Tomi, 2003. "*Finding Near Optimal Solutions for Vehicle Routing Problem with Time Windows Using Hybrid Genetic Algorithm*", http://www.unipa.it/Odysseus/Odysseus2003_file/ odysseus-main_file /pdf/AlvarengaMateus.pdf

Balas, E., dan N. Simonetti, 1996. "*Linear Time Dynamic-Programming Algorithms for New Classes of Restricted TSPs : A Computational Study*", http://www.extenza-eps.com/extenza/loadPDF?objectIDValue=9748

Braysy, O., 2002. "*A Fast Local Searches for The Vehicle Routing Problem with Time Windows*", INFOR, Vol. 40:4, 319-330.

Christine, 2004. "Study Tentang Vehicle Routing Problems dengan Menggunakan Standart Evolutionary", *Tugas Akhir Jurusan Teknik Industri,* No. 01/0766/IND/04, Universitas Kristen Petra, Surabaya.

Homberger, J., dan H. Gehring, 1999. "*Two Evolutionary Metaheuristics for The Vehicle Routing Problem with Time Windows*", INFOR, vol. 37, 297-318.

Kallehauge, B., J. Larsen, dan O.B.G. Marsen, 2001, "*Lagrangean Duality Applied on Vehicle Routing with Time Windows*", Technical Report, IMM, Technical University of Denmark.

Pesant, G., M. Gendreau, J-Y. Potvin, dan J-M. Rousseau, 1998. "*An Exact Constraint Logic Programming Algorithm for The Travelling Salesman Problem with Time Windows*", Transportation Science, Vol. 32, 12-29.

Prins, C., 2001, "*A Simple and Effective Evolutionary Algorithms for The Vehicle Routing Problem*", 4th Metaheuristics International Conference, 143-147.

Thangiah, S.R., 1995. "*Vehicle Routing with Time Windows Using Genetic Algorithms*", Application Handbook of Genetic Algorithms: New Frontiers, Vol. II, Lance Chambers (ed.), CRC Press, 253-277.