2. KAJIAN TENTANG *VEHICLE ROUTING PROBLEM*

Vehicle routing problem merupakan perkembangan atau perluasan dari permasalahan *Travelling Salesman Problem*. Terdapat berbagai macam variasi *vehicle routing problem,* dan beberapa macam variasinya akan dibahas pada bab ini.

2.1. *Travelling Salesman Problem* (TSP)

Egon Balas, dan Neil Simonetti mendefinisikan permasalahan TSP sebagai berikut : diberikan suatu *initial ordering* dari n kota, dan sebuah integer $k>0$, kemudian dicari biaya minimum dari *feasible tour*. *Feasible tour* yang dimaksud adalah *tour* dimana kota i didatangi sebelum kota j, jika $j \geq i + k$, dalam *initial ordering*. Jadi, pada permasalahan TSP yang dibahas oleh Egon Balas, dan Neil Simonetti diketahui N titik yang disebut kota, dan biaya perjalanan dari i ke j, c_{ij}, untuk semua $i, j \in N$. Tujuan dari TSP ini adalah mencari biaya permutasi atau *tour* minimum, dari semua kota. Jika $c_{ij} = c_{ji}$, maka permasalahan TSP tersebut simetris, jika tidak, maka disebut asimetris. TSP merupakan permasalahan mencari *Hamiltonian cycle* terpendek dalam G, dimana $G = (N, A)$ merupakan *directed* atau *undirected graph*, dengan panjang c_{ij} untuk semua $(i, j) \in A$.

Untuk menyelesaikan permasalahan TSP ini, Gilles Pesant, Michel Gendreau, Jean-Yves Potvin, dan Jean-Marc Rousseau memperkenalkan suatu *constraint proggramming model*. Parameter, dan variabel yang digunakan dalam model ini adalah :

- $V = \{2,...,n\}$ yang mewakili kota – kota yang akan dikunjungi.
- Untuk *origin-depot* dilambangkan dengan $V = 1$, sedangkan untuk *destination depot* dilambangkan dengan $V = n + 1$. Jadi, suatu *tour* akan merupakan *Hamiltonian path* yang berawal di 1 dan berakhir di $n + 1$.

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

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

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

- c_{ij} menunjukkan biaya *travel* dari kota *i* ke kota *j*.
- 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*.
- Sebuah *tour* yang *valid*, memberikan *distinct successor* pada tiap kota dan menghindari adanya *sub-tour*.
- b_i menunjukkan awal dari *partial part* yang melalui *i*, sedangkan e_i menunjukkan akhirnya; pada awalnya $b_i = e_i = i$.

Constraint proggramming model untuk TSP adalah :

- Fungsi tujuan :

$$Min \quad \sum_{i \in V^o} c_i, s_i \tag{2.1}$$

- Kendala :

$$S_i \neq S_j, \qquad \forall i, j \in V^o, i \neq j \tag{2.2}$$

$$S_i \neq i, \qquad \forall i \in V^o \tag{2.3}$$

$$S_i = j \Rightarrow S_{\in_j} \neq b_i, \ (\in_j \neq n+1) \qquad \forall i \in V^o \tag{2.4}$$

$$S_i \in \{2,..., n+1\}, \qquad \forall i \in V^o \tag{2.5}$$

Fungsi objektif dari permasalahan TSP adalah untuk meminimumkan total biaya dari *tour* : c_i , s_i yang merupakan biaya *travel* dari *i* ke *immediate successor Si* (2.1). Kendala (2.2), dan (2.5) memastikan bahwa tiap kota dikunjungi hanya satu kali saja, sedangkan kendala (2.3), dan (2.4) digunakan untuk menghilangkan *sub-tour*.

Kendala (2.5) hanya digunakan untuk menentukan *domain* . Kendala (2.3) digunakan untuk menghilangkan *i* dari *domain* pada tiap – tiap S_i. Kendala (2.2) menunggu sampai S_i atau S_j menjadi *fixed* pada nilai *k*, kemudian kendala ini akan menghilangkan *k* dari *domain* variabel lain. Pada kendala (2.4), 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, $\hat{I}_j$, tidak dapat diikuti oleh awalnya, b_i. Pada kasus khusus, jika $\hat{I}_j = n + 1$, yang berarti bahwa *path* sudah mencapai *destination depot*, maka tidak ada lagi yang perlu dilakukan.

2.1.1. *Travelling Salesman Problem with Time Windows* (TSPTW)

Gilles Pesant, Michel Gendreau, Jean-Yves Potvin dan Jean-Marc Rousseau, mendefinisikan permasalahan TSPTW sebagai permasalahan yang mencari biaya *tour* minimum 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* + waktu tunggu + waktu pelayanan). Jadi, untuk permasalahan TSPTW, terdapat tambahan kendala adanya *time windows* untuk masing – masing kota. *Time windows* $[a_i, b_i]$ menunjukkan batas waktu pelayanan di kota i, dimana a_i merupakan batas awalnya, dan b_i batas akhirnya. Kedatangan sebelum a_i diperbolehkan tetapi mengakibatkan adanya waktu tunggu sampai batas *time windows*, tetapi tidak diperbolehkan adanya kedatangan sesudah b_i.

Untuk permasalahan TSPTW ini, Gilles Pesant, Michel Gendreau, Jean-Yves Potvin dan Jean-Marc Rousseau, menambahkan beberapa parameter baru, yaitu :

- t_{ij} yang menunjukkan waktu *travel* antara kota i dan j.
- $T_i, i = 1,...,n+1$, yang menunjukkan waktu mulainya pelayanan di kota i.
- $T_1 = 0$

Jadi, untuk *constraint proggramming model* dari TSPTW, selain model (2.1) – (2.5), perlu ditambahkan kendala sebagai berikut :

$$a_i \le T_i \le b_i, \qquad \forall i \in V^d \qquad (2.6)$$

$$Si = j \Rightarrow T_i + t_{ij} \le T_j \qquad \forall i \in V^o \qquad (2.7)$$

Kendala (2.6) membatasi agar waktu pelayanannya sesuai dengan *time windows*-nya, sedangkan kendala (2.7) untuk memastikan feasibilitas dari jadwal yang dibuat.

2.2. *Vehicle Routing Problem* (VRP)

Brian Kallehauge, Jesper Larsen, dan Oli B.G. Marsen (2001) mendefinisikan permasalahan *m*-TSP sebagai salah satu variasi dari TSP, dimana terdapat *m-salesman* yang harus 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.

VRP merupakan permasalahan *m*-TSP, dimana sebuah *demand* dapat diasosiasikan dengan sebuah kota atau seorang konsumen, dan tiap kendaraan memiliki kapasitas tertentu. Total jumlah *demand* dalam suatu rute, tidak boleh melebihi kapasitas dari kendaraan yang ditugasi 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).

Christian 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*, yang mempunyai 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 minimum, 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 Christian Prins, jumlah kendaraan adalah variabel keputusan, dan biayanya adalah bilangan *real.*

Sam R. Tangiah membuat suatu *mixed-integer formulation* dari permasalahan VRP ini. Parameter dan variabel yang digunakan dalam *mixed-integer formulation* tersebut antara lain :

- K = nomer kendaraan
- N = nomer konsumen (0 menunjukkan $depot$)
- C_i = konsumen i
- C_0 = $depot$
- V_k = rute kendaraan k
- c_{ijk} = biaya *travel* antara konsumen i dan j untuk kendaraan k.
- q_{ik} = total *demand* kendaraan k sampai konsumen i.
- v_k = kapasitas maksimum kendaraan k
- $y_{ik} = \begin{cases} 1, \text{ jika konsumen } i \text{ dilayani oleh kendaraan } k \\ 0, \text{ jika tidak} \end{cases}$
- $x_{ijk} = \begin{cases} 1, \text{ jika kendaraan } k \text{ dari konsumen } i \text{ langsung ke konsumen } j \\ 0, \text{ jika tidak} \end{cases}$

Mixed-integer formulation untuk permasalahan VRP ini adalah :

- Fungsi tujuan :

$$Min \sum_{i=0}^{N} \sum_{j=0}^{N} \sum_{k=1}^{K} c_{ij} x_{ijk} \tag{2.8}$$

- Kendala :

$$\sum_{i=0}^{N} q_{ik} y_{ik} \le v_k \ , k = 1,... \ , K \tag{2.9}$$

$$y_{ik} = 0 \text{ atau } 1 \ ; \qquad i = 1, 2, \ldots, N \ ; \qquad k = 1, 2, \ldots, K \tag{2.10}$$

$$x_{ijk} = 0 \text{ atau } 1 \ ; \qquad i = 1, 2, \ldots, N \ ; \qquad k = 1, 2, \ldots, K \tag{2.11}$$

$$\sum_{k=1}^{K} y_{ik} = \begin{cases} K, i = 0 \\ 1, i = 1, ..., N \end{cases} \tag{2.12}$$

$$\sum_{i=0}^{N} x_{ijk} = y_{jk} \ , j = 0, ..., N; \ k = 1, ..., K \tag{2.13}$$

$$\sum_{j=0}^{N} x_{ijk} = y_{ik} \ , i = 1, ..., N; \ k = 1, ..., K \tag{2.14}$$

Tujuan dari permasalahan VRP ini adalah untuk meminimalkan total biaya *travel.* Kendala (2.9) membatasi bahwa total jumlah *demand* yang dibawa oleh kendaraan k, tidak boleh melebihi kapasitas dari kendaraan tersebut. Kendala (2.12) dapat digunakan untuk menunjukkan bahwa tiap konsumen hanya dapat dilayani oleh satu kendaraan saja. Kendala (2.13), dan (2.14) digunakan untuk memastikan bahwa tiap konsumen dikunjungi oleh kendaraan yang sama dengan yang sudah dijadwalkan untuk konsumen tersebut.

Brian Kallehauge, Jesper Larsen, dan Oli B.G. Marsen (2001), memodelkan permasalahan VRP sebagai berikut :

- Fungsi tujuan :

$$\text{Min} \sum_{k=1}^{K}\sum_{i=0}^{N+1}\sum_{j=0}^{N+1} c_{ij}x_{ijk} \qquad (2.15)$$

- Kendala :

$$\sum_{k=1}^{K}\sum_{j=0}^{N+1} x_{ijk} = 1; \qquad i = 1, 2, ..., N \qquad (2.16)$$

$$\sum_{i=1}^{N} d_i \sum_{j=0}^{N+1} x_{ijk} \le v_k; \qquad k = 1, 2, ..., K \qquad (2.17)$$

$$\sum_{j=0}^{N+1} x_{ojk} = 1; \qquad k = 1, 2, ..., K \qquad (2.18)$$

$$\sum_{i=0}^{N+1} x_{ihk} - \sum_{j=0}^{N+1} x_{hjk} = 0; \qquad h = 1, 2, ..., N \; ; k = 1, 2, ..., K \qquad (2.19)$$

$$\sum_{i=0}^{N+1} x_{i,N+1,k} = 1; \qquad k = 1, 2, ..., K \qquad (2.20)$$

$$x_{ijk} \in \{0,1\} \quad ; \qquad i = 0, 2, \ldots, N + 1 \; ; k = 1, 2, \ldots, K \qquad (2.21)$$

Untuk model matematika yang dibuat oleh Brian Kallehauge, Jesper Larsen, dan Oli B.G. Marsen (2001) ini, terdapat beberapa perubahan atau penyesuaian dari model yang terdapat pada jurnal aslinya. Hal ini dilakukan, agar dapat mengkaitkan model tersebut dengan model yang dibuat oleh Sam R. Tangiah. Pada model yang dibuat oleh Brian Kallehauge, Jesper Larsen, dan Oli B.G. Marsen (2001) ini, terdapat tambahan parameter d_i yang menunjukkan jumlah *demand* dari konsumen *i.*

Fungsi tujuan dari model yang dibuat oleh Brian Kallehauge, Jesper Larsen, dan Oli B.G. Marsen (2001), sama seperti fungsi tujuan dari model yang dibuat oleh Sam R. Tangiah, yaitu untuk meminimalkan total biaya *travel*. Tetapi, Sam R. Tangiah 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 Brian Kallehauge, Jesper Larsen, dan Oli B.G. Marsen (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 (2.16) mempunyai kegunaan yang sama seperti kendala (2.12), yaitu untuk menunjukkan bahwa tiap konsumen hanya dapat dilayani oleh satu kendaraan saja. Kendala (2.17) mempunyai kegunaan yang sama seperti kendala (2.9), yaitu untuk membatasi bahwa total jumlah *demand* yang dibawa oleh kendaraan k, tidak boleh melebihi kapasitas dari kendaraan tersebut. Kendala (2.18) – (2.20) 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 Sam R. Tangiah, tidak terdapat kendala yang mempunyai kegunaan seperti kendala (2.18) – (2.20) ini.

2.2.1. *Vehicle Routing Problem with Time Windows* (VRPTW)

Brian Kallehauge, Jesper Larsen, dan Oli B.G. Marsen (2001), mendefinisikan permasalahan VRPTW sebagai perluasan dari permasalahan VRP. Jika pada permasalahan VRP ditambahkan adanya *time window* dari masing – masing konsumen, maka permasalahan tersebut menjdai permasalahan VRPTW. Untuk permasalahan 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"*.

Sam R. Tangiah mendefinisikan VRPTW sebagai permasalahan yang menjadwalkan sekumpulan kendaraan, dengan kapasitas dan *travel time* yang terbatas, dari *central depot* ke sekumpulan konsumen yang tersebar secara geografis, dengan *demand* yang 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.

Jorg Homberger dan Hermann Gehring 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 adalah untuk menentukan jadwal rute yang *feasible*, dengan pertama, meminimumkan jumlah kendaraan, dan kedua, meminimumkan 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 mengenai permasalahan VRPTW diatas, membahas mengenai permasalahan VRPTW dengan *hard time windows*. Untuk permasalahan *hard time windows*, 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 permasalahan VRPTW ini, Sam R. Tangiah memperkenalkan beberapa parameter yang perlu ditambahkan, yaitu :

- R_k = total waktu rute untuk kendaraan k.
- t_{ij} = waktu *travel* antara konsumen i dan j (proporsional dengan jarak *Euclidean*)
- t_i = waktu kedatangan di konsumen i
- f_i = waktu *service* di konsumen i
- w_i = waktu tunggu sebelum melayani konsumen i
- $w_i = max\ \{\ 0\ ,\ (\ e_i - t_i\ )\ \}$
- e_i = waktu paling awal untuk pelayanan di konsumen i
- l_i = waktu paling akhir untuk pelayanan di konsumen i

VRPTW mempunyai fungsi tujuan yang sama dengan persoalan VRP, yaitu untuk meminimumkan total biaya *travel* semua kendaraan (2.8). Sedangkan untuk kendalanya juga sama dengan VRP [(2.9)-(2.14)], tetapi perlu ditambahkan beberapa kendala lagi yang berhubungan dengan kendala *time windows*. Kendala – kendala yang perlu ditambahkan tersebut adalah:

$$\sum_{i=0}^{N}\sum_{i=0}^{N} y_{ik}(t_{ij} + f_i + w_i) \le R_k\ ; \qquad k = 1, ..., K \tag{2.22}$$

$$t_j \ge t_i + w_i + f_i + t_{ij} - M(1 - x_{ijk})\ ; \quad i,j = 1,...,N;\ k = 1,..., K \tag{2.23}$$

$$e_i \le t_i < l_i\ ; \qquad \text{i} = 1, ..., \text{N} \tag{2.24}$$

$$t_i \ge 0\ ; \qquad \text{i} = 1, ..., \text{N} \tag{2.25}$$

Kendala (2.22) digunakan untuk memastikan bahwa tiap kendaraan melayani semua konsumen yang dijadwalkan untuk kendaraan tersebut, tanpa melebihi waktu *travel* dari kendaraan. Kendala (2.23) digunakan untuk memastikan bahwa waktu kedatangan dari kedua konsumen adalah *compatible*. *M*

merupakan bilangan yang sangat besar. Kendala (2.24) mengharuskan kendaraan untuk sampai di tiap – tiap konsumen selama batas *time window* dari konsumen tersebut. Kendala (2.25) memastikan bahwa waktu kedatangan kendaraan ke tiap konsumen selalu positif.

Untuk memodelkan permasalahan VRPTW, Brian Kallehauge, Jesper Larsen, dan Oli B.G. Marsen (2001), juga menambahkan beberapa kendala ke model permasalahan VRP sebelumnya, yaitu yang terdapat pada persamaan [(2.15) – (2.21)]. Kendala yang ditambahkan tersebut adalah :

$$t_i + t_{ij} - M(1 - x_{ijk}) \leq t_j \; ; \quad i, j = 0, 1, ..., N+1 \; ; k = 1, 2, ..., K \qquad (2.26)$$

$$e_i \leq t_i \leq l_i \; ; \qquad i = 0, 1, ..., N+1 \qquad (2.27)$$

Parameter t_{ij} yang digunakan pada model ini memiliki arti yang berbeda dari parameter t_{ij} yang digunakan pada model yang dibuat oleh Sam R. Tangiah. Pada model yang dibuat oleh Brian Kallehauge, Jesper Larsen, dan Oli B.G. Marsen (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. Untuk model dari kedua kendala tambahan yang dibuat oleh Brian Kallehauge, Jesper Larsen, dan Oli B.G. Marsen (2001) ini, terdapat beberapa perubahan atau penyesuaian dari model yang terdapat pada jurnal aslinya. Hal ini dilakukan, agar dapat mengkaitkan model tersebut dengan model yang dibuat oleh Sam R. Tangiah.

Kendala (2.26) mirip dengan kendala (2.23), dan juga mempunyai kegunaan yang sama dengan kendala (2.23), yaitu untuk memastikan bahwa waktu kedatangan dari kedua konsumen adalah *compatible*. Perbedaaan antara kendala (2.23), dan (2.26) adalah, bahwa kendala (2.26) mengikut sertakan *depot*, sedangkan kendala (2.23) hanya untuk konsumen saja, tanpa mengikut sertakan *depot*. Sedangkan kendala (2.27) juga mirip dengan kendala (2.24), dan juga mempunyai kegunaan yang sama dengan kendala (2.24), yaitu untuk mengharuskan kendaraan untuk sampai di tiap – tiap konsumen selama batas *time window* dari konsumen tersebut. Perbedaaan antara kendala (2.24), dan (2.27) adalah, bahwa kendala (2.27) mengikut sertakan *depot*, sedangkan kendala (2.24) hanya untuk konsumen saja, tanpa mengikut sertakan *depot*.

Untuk menyelesaikan permasalahan VRPTW ini, Olli Braysy menyarankan suatu algoritma yang berdasarkan pada pendekatan tiga fase. Fase pertama adalah *initial solution* yang dibuat dengan 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 meminimumkan total jarak yang ditempuh untuk tiap rute. Urutan langkah dari algoritma tersebut adalah sebagai berikut :

a. Membuat *initial solution* dengan menggunakan *Hybrid Construction* atau *Merge Heuristic*.
b. Mengulangi prosedur pengurangan rute, sampai tidak ada lagi rute yang dapat dihilangkan.
c. Mengulangi langkah pertama, dan kedua, dengan menggunakan semua nilai parameter dalam batas limit tertentu. Hasil solusi yang dibuat tersebut kemudian disimpan.
d. Dari semua hasil solusi yang telah diperoleh, dicari solusi dengan jumlah rute paling sedikit, dan dimasukkan dalam set *RB.*
e. Menata ulang urutan dalam solusi S_i, menurut parameter b, dan memperbaiki S_i dengan menggunakan prosedur *Or-opt*.
f. Mengulang langkah kelima untuk semua S_i dalam *RB* dan meng-*update* solusi terbaik yang ditemukan, S_b, jika dibutuhkan.
g. 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 Olli Braysy 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.

G.B. Alvarenga, G.R. Mateus, dan G. De Tomi, mengusulkan suatu pendekatan *robust heuristic* untuk permasalahan VRPTW, dengan menggunakan *Genetic Algorithm* yang efisien dan formulasi 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)*, yang telah sering digunakan pada penelitian sebelumnya. Pada tahap *selection*, dipilih sepasang *parents* untuk *crossover.* Pada penelitian ini, 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*.

Pada penelitian ini digunakan metode *crossover* dengan langkah – langkah sebagai berikut :

a. Pada langkah pertama, dibuat sebuah pilihan rute secara *random* untuk tiap *parent individual.*
b. Setelah semua rute yang *feasible* dimasukkan, untuk tiap konsumen yang tertinggal, akan diuji apakah dapat disisipkan pada rute yang tidak kosong dari individu baru.
c. 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 :

a. *Random Customer Migration*
b. *Bringing the Best Customer*
c. *Re-insertion using PFIH*
d. *Similar Customer Exchange*
e. *Exchanging Customer with Positive Gain*
f. *Merging Two Routes*

g. *Reinserting Customer*

h. *Route Partitioning*

Algoritma yang diusulkan oleh G.B. Alvarenga, G.R. Mateus, dan G. De Tomi, 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.

2.2.2. *Vehicle Routing Problem with General Time Window Constraint* (VRPGTW)

Thoshihide Ibaraki, Mikio Kubo, Tomoyasu Masuda, Takeaki Uno, dan Mutsunori Yayiura (2001), membahas mengenai permasalahan VRP dengan *soft time windows*. Untuk permasalahan *soft time windows*, batas *time windows* dan kapasitas kendaraan dapat tidak dipatuhi. Apabila terjadi pelanggaran terhadap batas *time windows* dan kapasitas kendaraan tersebut, maka akan dikenakan suatu penalti. Permasalahan VRP dengan *soft time windows* sering disebut sebagai *Vehicle Routing Problem with General Time Window Constraint* (VRPGTW).

Untuk permasalahan VRPGTW, setelah menentukan rute yang harus dilalui oleh kendaraan, harus ditentukan pula waktu awal pelayanan untuk semua konsumen yang optimal, agar total penalti yang diperoleh minimum. Jadi, tujuan dari permasalahan VRPGTW adalah meminimumkan { total jarak *travel* + total penalti karena datang sebelum waktu pelayanan dimulai + total barang yang melebihi kapasitas kendaraan }.

T. Ibaraki, M. Kubo, T. Masuda, T. Uno, dan M Yagiura (2001), mendefinisikan permasalahan VRPGTW sebagai berikut : $G = (V, E)$ adalah sebuah *complete directed graph* dengan sebuah *vertex set* $V = \{ 0, 1, ..., n \}$, dan sebuah *edge set* $E = \{(i, j) \mid i, j \in V, i \neq j \}$, dan $M = \{ 1, 2, ..., m \}$ adalah sekumpulan kendaraan. *Vertex 0* adalah *depot*, dan *vertex* lainnya adalah konsumen. Parameter – parameter yang berhubungan dengan tiap konsumen $i \in V$, tiap kendaraan $k \in M$, dan tiap $edge (i, j) \in E$, adalah :

- *Demand* q_i (≥ 0) yang harus dikirimkan.
- *Penalty function* $p_i(t)$ (≥ 0), dari *start time service t*.
- Waktu *service* u_i (≥ 0).
- Kapasitas Q_k (≥ 0).
- Jarak d_{ij} (≥ 0).
- Waktu *travel* t_{ij} (≥ 0).

Diasumsikan bahwa $q_0 = 0$, dan $u_0 = 0$, untuk *depot 0*. Tiap kendaraan dapat berangkat dari *depot* setelah waktu *0*. Tiap *penalty function* $p_i(t)$ adalah nonnegatif, dan memenuhi $p_i(t) \leq \lim_{e \to 0} \min\{p_i(t+e), p_i(t-e)\}$, pada tiap *discontinuous point t*.

s_k menunjukkan rute yang dilayani oleh kendaraan k, dimana $s_k(h)$ menunjukkan konsumen ke h dalam s_k, dan $s = (s_1, s_2, ..., s_m)$. Tiap konsumen i dimasukkan pada tepat satu s_k, dan dikunjungi oleh kendaraan tepat satu kali. n_k menunjukkan jumlah konsumen dalam s_k, untuk $k \in M$. Karena tiap kendaraan $k \in M$, memulai dan mengakiri perjalanannya di *depot*, maka dapat dituliskan bahwa $s_k(0) = 0$, dan $s_k(n_k + 1) = 0$, untuk semua k. s_i adalah waktu mulainya pelayanan di konsumen i, sedangkan s_i^a adalah waktu kedatangan kendaraan di konsumen i.

Variabel yang digunakan adalah $y_{ik}(s)$, dimana $y_{ik}(s) \in \{0, 1\}$, untuk $i \in V$, dan $k \in M$. $y_{ik}(s) = 1$, jika dan hanya jika kendaraan k mengunjungi konsumen i tepat satu kali. Total jarak yang ditempuh oleh semua kendaraan $\{d_{sum}(s)\}$, total *penalty* untuk waktu mulainya *service* $\{p_{sum}(s)\}$, dan total dari kelebihan kapasitas $\{q_{sum}(s)\}$, dapat dituliskan sebagai berikut :

$$d_{sum}(s) = \sum_{k \in M} \sum_{h=0}^{n_k} d_{s_k(h), s_k(h+1)}$$

$$p_{sum}(s) = \sum_{i \in V \setminus \{0\}} p_i(s_i) + \sum_{k \in M} p_0(s_k^a)$$

$$q_{sum}(s) = \sum_{k \in M} \max\left\{\sum_{i \in V} q_i y_{ik}(s) - Q_k, 0\right\}$$

Model dari permasalahan VRPGTW adalah sebagai berikut :

- Fungsi tujuan :

$$Min \cos t(s, s) = d_{sum}(s) + p_{sum}(s) + q_{sum}(s) \quad (2.28)$$

- Kendala :

$$\sum_{k \in M} y_{ik}(s) = 1; \qquad i \in V \setminus \{0\} \quad (2.29)$$

$$t_{0,s_k(1)} \le s_{s_k(1)}; \qquad k \in M \quad (2.30)$$

$$s_{s_k(h)} + u_{s_k(h)} + t_{s_k(h),s_k(h+1)} \le s_{s_k(h+1)}; \quad h = 1, 2, ..., n_k\text{-}1; k \in M \quad (2.31)$$

$$s_{s_k(n_k)} + u_{s_k(n_k)} + t_{s_k(n_k),0} \le s_k^a; \qquad k \in M \quad (2.32)$$

Kendala (2.29) digunakan untuk memastikan bahwa tiap konsumen $i \in V \setminus \{0\}$, harus dilayani hanya satu kali saja oleh tepat satu kendaraan. Kendala (2.30), (2.31), dan kendala (2.32) digunakan untuk memastikan bahwa waktu mulainya service di kosnumen *i*, s_i, harus setelah waktu kedatangan kendaraan di konsumen *i*. Jadi, pada permasalahan VRPGTW, solusi yang *feasible* adalah solusi yang memenuhi kendala (2.29) – (2.32), tanpa perlu memenuhi kendala *time windows* dan kendala kapasitas yang ada.

Untuk menyelesaikan permasalahan VRPGTW ini, T. Ibaraki, M. Kubo, T. Masuda, T. Uno, dan M Yagiura (2001), memperkenalkan metode penyelesaian dengan menggunakan *local search*. *Local search* tersebut digunakan untuk menjadwalkan konsumen ke kendaraan, dan untuk mencari urutan konsumen yang harus dikunjungi oleh kendaraan. Untuk VRP, selain *standart neighborhoods* yang ada, dibutuhkan sebuah *neighborhoods* baru yang disebut *cyclic exchange neighborhood*. Setelah menetapkan urutan dari konsumen yang harus dikunjungi oleh kendaraan, harus ditentukan pula *optimal start time* untuk melakukan kunjungan ke konsumen, agar total penalti yang diperoleh minimum. Untuk menentukan *start time* yang optimal ini, digunakan *dynamic programming.*

Yang dimaksud dengan *standart neighborhoods*, adalah *cross exchange*, *2-opt*, dan *Or-opt neighborhoods*. Sedangkan yang dimaksud dengan *cyclic exchange neighborhood* adalah sekumpulan solusi yang diperoleh dengan mengubah secara berkala, dua atau lebih, panjang *paths*, sampai maksimal L^{cyclic} (sebuah parameter).

Metaheuristics yang berdasar pada *local search*, yang digunakan pada penelitian ini, adalah *multi-start local search* (MLS), *iterated local search* (ILS), dan *adaptive multi-start local search* (AMLS). MLS secara berulang mengaplikasikan *local search* (LS) pada sejumlah *initial solution* yang dibangkitkan secara random atau dengan menggunakan metode *greedy*, dan outputnya adalah solusi terbaik yang ditemukan selama keseluruhan pencarian. ILS merupakan variasi dari MLS, dimana *initial solution* untuk LS dibangkitkan dengan *perturbing* solusi baik, yang ditemukan dari pencarian sebelumnya. AMLS juga merupakan salah satu variasi dari MLS, yang menyimpan sejumlah *P* solusi baik, yang ditemukan pada pencarian sebelumnya, dan membangkitkan *initial solution* dengan mengkombinasikan bagian dari solusi pada *P*.

Metode ini kemudian diujikan untuk tiga macam permasalahan, yaitu permasalahan *Solomon's VRPHTW benchmark*, permasalahan penjadwalan mesin paralel dengan beraneka macam *time windows*, dan permasalahan dunia nyata yang menyangkut penjadwalan produksi dengan biaya inventori. Dari hasil perhitungan terlihat bahwa ILS dan AMLS memberikan prospek yang bagus. Untuk permasalahan *Solomon's VRPHTW benchmark*, metode yang digunakan ini mampu menghasilkan solusi yang lebih baik dari hasil terbaik yang telah ditemukan sebelumnya, untuk 14 jenis permasalahan, dan mampu menghasilkan solusi yang sama untuk 3 permasalahan, dari 39 jenis permasalahan yang diujikan. Untuk permasalahan penjadwalan mesin paralel, dibuat permasalahan dengan *time penalty function* (termasuk yang *non-convex*), yang hasil optimalnya diketahui. Dari hasil pengujian, diketahui bahwa algoritma yang diusulkan pada penelitian ini, kemungkinan besar mampu menghasilkan solusi yang optimal ataupun mendekati optimal. Dari hasil pengujian yang dilakukan juga dapat disimpulkan bahwa *cyclic exchange neighborhood* lumayan efektif untuk permasalahan dengan *non-convex time penalty function*.

2.2.3. *Fleet Size and Mix Vehicle Routing Problem* (FSMVRP)

Untuk permasalahan VRP yang telah dibahas sebelumnya, jenis kendaraan yang digunakan adalah *homogeneous*. Kendaraan disebut *homogeneous*

bila semua kendaraan memiliki kapasitas, *fixed utilization costs* dan *variabel travelling cost* yang sama. Tetapi pada kenyataannya, semua perusahaan belum tentu memiliki kendaraan yang *homogeneous*. Oleh karena itu, muncul permasalahan VRP yang menggunakan kendaraan *heterogeneous*. Permasalahan tersebut sering disebut sebagai *Mix Fleet Vehicle Routing Problem* (MFVRP), *Fleet Size and Mix Vehicle Routing Problem* (FSMVRP), *Vehicle Fleet Mix Problem* atau *Fleet Size and Composition Vehicle Routing Problem*.

Devern Burchett, dan Edward Campion (2002), membahas mengenai permasalahan MFVRP, yang merupakan salah satu variasi dari permasalahan VRP. MFVRP merupakan pengembangan dari VRP, yang memasukkan *vehicle fleet composition decision*, dengan menggunakan sejumlah kendaraan *heterogeneous*. Tiap kendaraan memiliki kapasitas, *fixed cost*, dan *variable travel cost* yang berbeda – beda. Tiap rute berawal dan berakhir di *depot*, dan tujuannya adalah meminimumkan biaya *service* semua konsumen. Biaya ini merupakan penjumlahan dari semua *fixed costs*, dan *variable cost* dari jarak yang ditempuh oleh masing – masing kendaraan.

Jacques Renaud dan Fayes F. Boctor, membahas mengenai permasalahan FSMVRP. Permasalahan FSMVRP menyangkut dua hal yang utama, yaitu komposisi dari kendaraan *heterogeneous* yang hendak digunakan dan juga rute dari tiap – tiap kendaraan tersebut. Tiap kendaraan memiliki kapasitas, dan biaya tetap, serta biaya variabel yang berbeda – beda. Tujuan dari FSMVRP adalah untuk meminimumkan total biaya, yang terdiri dari *fixed utilization cost* dan *variable travelling cost* dari kendaraan. Tujuan tersebut dapat dicapai dengan mencari komposisi kendaraan yang optimal, dan juga dengan menentukan rute untuk tiap kendaraan tersebut yang memenuhi semua kendala yang ada.

Jacques Renaud, dan Fayez F. Boctor, mendefinisikan permasalahan FSMVRP sebagai berikut : $G = (V, A)$, adalah suatu *graph* dimana $V = \{v_0, ..., v_n\}$, adalah sekumpulan *vertex*, dan $A = \{ (v_i, v_j) : v_i, v_j \in V, i \neq j \}$, adalah sekumpulan *arc*. *Vertex* v_0 menunjukkan suatu *depot*, yang memiliki M kendaraan dengan tipe yang berbeda. Tiap *vertex* $v_i \in V \setminus \{ v_0 \}$, berkaitan dengan seorang konsumen, yang memiliki *demand* non-negatif q_i, dan waktu *service* s_i. Tiap *edge* (v_i, v_j) berhubungan dengan biaya non-negatif, c_{ij}, yang menunjukkan biaya

travel-nya, dan waktu non-negatif, t_{ij}, yang menunjukkan lama *travel*-nya. Sebagai tambahan, F_k, Q_k, dan T_k menunjukkan biaya tetap kendaraan, kapasitas kendaraan, dan maksimum waktu *travel* untuk kendaraan tipe $k = 1, ..., M$. Tujuan dari permasalahan FSMVRP adalah untuk menentukan sekumpulan kendaraan yang berbeda tipe yang akan digunakan, dan juga rute untuk tiap kendaraan tersebut, dengan memenuhi semua kendala yang ada. Kendala tersebut adalah :

- Rute kendaraan berawal dan berakhir di *depot.*
- Tiap konsumen dikunjungi tepat satu kali.
- Total *demand* dari suatu rute tidak melebihi kapasitas dari kendaraan yang dijadwalkan untuk rute tersebut.
- Total *duration* untuk tiap rute (termasuk waktu *travel* dan service) tidak melebihi maksimum waktu *travel* T_k dari kendaraan yang digunakan.
- Jumlah dari biaya tetap dan variabelnya minimum.

Jacques Renaud, dan Fayez F. Boctor memperkenalkan pemakaian *Sweep Based Algorithm* untuk menyelesaikan permasalahan FSMVRP tersebut. Heuristik yang diusulkan tersebut menggunakan prosedur yang berbeda untuk membangkitkan sejumlah besar rute bagus, dan kemudian memilih rute yang memenuhi kendala dari permasalahan, dengan biaya yang paling minimum, menggunakan *polynomial set partitioning algorithm.* Secara khusus, heuristik yang diusulkan ini menggunakan lima prosedur subordinatif yang disebut : *Order, 1-petal, 2-petal, Petals Selection,* dan *Improve.*

Langkah - langkah dari heuristik yang diusulkan ini adalah sebagai berikut :

a. *Orders determination*
b. *Order selection*
c. *Route initialization*
d. *1-Petal route*
e. *2-Petal route*
f. *Dominance test*
g. *Petals selection*
h. *Improvement of the initial solution*

Dari 20 *benchmark test problems*, versi standart dari *sweep heuristic* yang diusulkan, menghasilkan rata – rata deviasi diatas *best-known solution*, sebesar 0.49 % dengan rata – rata waktu menghitung sebesar 179 detik. Heuristik yang diusulkan ini menghasilkan tujuh *best-known solution*, dan tiga diantaranya merupakan *best-known solution* baru. Versi yang lebih cepat dari algoritma ini, juga diusulkan oleh Jacques Renaud, dan Fayez F. Boctor. Algoritma versi tersebut menghasilkan rata – rata deviasi sebesar 0.63 % diatas *best-known solution*, dengan rata – rata waktu menghitung selama 154 detik.

2.2.4. *Period Vehicle Routing Problem* (PVRP)

Dalessandro Soares Vianna, Luiz S. Ochi, dan Lucia M. A. Drummond membahas mengenai Period Vehicle Routing Problems (PVRP). PVRP merupakan perluasan dari permasalahan VRP yang klasik. Dalam permasalahan VRP yang klasik, *planning period* yang digunakan hanya satu hari, sedangkan dalam PVRP, *planning period*-nya diperpanjang menjadi *M* hari. Dalam jangka waktu *M* hari tersebut, tiap konsumen harus dikunjungi minimal sekali. Permasalahan PVRP yang klasik terdiri dari kendaraan *homogeneous* (kendaraan dengan kapasitas yang sama), yang harus mengunjungi sekelompok konsumen dari sebuah *depot*, dimana kendaraan tersebut harus mengawali dan mengakhiri rute perjalanannya di *depot* tersebut. Tiap kendaraan memiliki kapasitas yang tetap, yang tidak boleh dilampaui, dan tiap konsumen memiliki *demand* harian yang harus dipenuhi dalam sekali kunjungan saja, oleh satu kendaraan. *Planning period*-nya adalah *M* hari. Jika $M = 1$, maka permasalahan PVRP tersebut menjadi permasalahan VRP. Untuk permasalahan PVRP, tiap konsumen harus dikunjungi sebanyak *k* kali, dimana $1 \leq k \leq M$. Pada permasalahan PVRP yang klasik, *demand* harian dari tiap konsumen selalu tetap. Permasalahan PVRP dapat dilihat sebagai permasalahan membentuk suatu *group* rute untuk tiap hari, agar kendala yang ada dapat dipenuhi, dan biaya globalnya dapat minimum.

PVRP dapat dilihat sebagai *multi-level combinatorial optimization problem*. Pada level pertama, tujuannya adalah untuk menentukan sekelompok alternatif yang *feasible* untuk tiap konsumen. Contohnya, bila $M = 3$, maka

kombinasi yang mungkin, adalah : *000, 001, 010, 100, 011, 110, 101, 111*. Jika konsumen menghendaki hanya ada dua kunjungan, maka alternatif kunjungan tersebut adalah : (d_1,d_2), (d_1,d_3), atau (d_2,d_3). Pada level kedua, harus ditentukan alternatif mana yang akan digunakan pada konsumen tersebut agar kendala hariannya terpenuhi. Sehingga harus ditentukan konsumen mana saja yang akan dilayani untuk tiap harinya selama *M* hari tersebut. Pada level ketiga, permasalahan VRP untuk tiap hari tersebut diselesaikan.

Untuk mengelesaikan permasalahan PVRP ini, Dalessandro Soares Vianna, Luis S.Ochi, dan Lucia M.A. Drummond memperkenalkan suatu metode yang menggunakan *Parallel Hybrid Evolutionary Metaheuristics* (PAR-HEM). Walaupun tujuan dari PAR-HEM adalah untuk mengurangi waktu yang diperlukan untuk mendapat solusi yang dapat diterima, kadang – kadang PAR-HEM juga dapat digunakan untuk memperbaiki hasil yang didapatkan oleh sequential versions. *Parallel Hybrid Evolutionary Metaheuristics* yang diusulkan oleh Dalessandro Soares Vianna, Luis S.Ochi, dan Lucia M.A. Drummond, adalah PAR-HEM yang berdasarkan pada paralel GA, *scatter search*, dan metode *local search*.

Algoritma paralel berdasarkan pada *Island Model*. Dalam *Island Model*, populasi dari *chromosome* GA di bagi menjadi beberapa sub populasi, yang berevolusi secara paralel, dan secara berkala, individu (*chromosome*) melakukan migrasi diantara mereka sendiri. Frekuensi terjadinya migrasi ini tidak terlalu sering. Jadi, migrasi hanya dilakukan saat pembaharuan sub populasi diperlukan. Kriteria penghentian proses berdasar pada kondisi global, yang berhubungan dengan semua proses yang melakukan PAR-HEM, dengan tujuan untuk mencegah *idle* dari sebuah proses, saat proses lainnya sedang berjalan.

Chromosome yang digunakan pada GA, ditampilkan oleh dua vektor. Vektor pertama berhubungan dengan *n* konsumen dalam PVRP. Posisi ke *i* dari vektor ini adalah integer non-negatif *k*, sehingga $0 \leq k \leq 2t-1$, dimana *t* adalah jumlah hari dalam *planning horizon* yang digunakan. Nomer *k* menunjukkan alternatif kunjungan yang *feasible* (kombinasi), ke konsumen yang terkait. Penyampaian *k* dalam bentuk *binary* menunjukkan hari dimana konsumen tersebut akan menerima kunjungan. Vektor kedua (berhubungan dengan yang pertama),

berkaitan dengan akumulasi jumlah *demand* untuk tiap hari pada *planning horizon*.

Untuk membuat *initial popula tion* dari GA, pertama kali, dibangkitkan sekumpulan alternatif kunjungan yang *feasible* untuk tiap konsumen. Konsumen diurutkan sesuai dengan jumlah alternatif mereka. Untuk membangkitkan sebuah *initial chromosome*, secara random dipilih sebuah konsumen dari sekumpulan konsumen yang memiliki jumlah alternatif *feasible* yang paling kecil. Kemudian, salah satu alternatif dari konsumen ini dipilih. Nomer integer yang berhubungan dengan alternatif ini ditempatkan pada *gene* yang berkorespondensi. Kemudian, sebuah alternatif dipilih dari tiap konsumen dengan jumlah alternatif paling sedikit yang masih tertinggal. Setiap sebuah alternatif dipilih, data pada vektor kedua perlu di-*update*. Jika penambahan alternatif dari konsumen baru tersebut melanggar kapasitas global dari kendaraan pada hari manapun, maka alternatif ini dibuang dan digantikan dengan alternatif lain miliki konsumen baru tersebut. Saat konsumen tidak memiliki alternatif lainnya, *chromosome* tersebut ditinggalkan. *Chromosome* disebut lengkap apabila setiap *gene* telah terisi.

Mekanisme repdoduksi pada GA menggunakan *classical crossover* dan *mutation operator*. Pada *crossover operator*, dari dua *parents* yang diambil dari populasi, dipilih secara random sebuah posisi (*gene*) yang berkaitan dengan seorang konsumen. Pada awalnya, dari vektor kedua, diambil nilai *demand* yang berkaitan dengan alternatif yang dipilih tersebut. Kemudian, dicoba untuk mengganti alternatif konsumen, yang dipilih dari tiap *parent* tersebut (jika alternatif ini sangat berbeda). Jika pergantian ini tidak menghasilkan solusi baru yang *feasible*, berkaitan dengan kendala dari kapasitas maksimum kendaraan tiap harinya, maka akan dilakukan percobaan lain terhadap *gen*e yang baru. Prosedur ini dilakukan *p* kali, dimana *p* adalah sebuah input paramater dari permasalahan. Operator kedua yang digunakan adalah *Mutation operator*, yang melakukan perubahan secara *feasible* dalam *p point* yang dipilih secara random dalam *chromosome*.

Algoritma yang diperkenalkan oleh Dalessandro Soares Vianna, Luis S.Ochi, dan Lucia M.A. Drummond ini, diaplikasikan pada beberapa permasalahan yang dipresentasikan oleh Christofides dan Beasley, serta pada

permasalahan yang dipresentasikan oleh Golden, Chao, dan Wasil. Permasalahan ini memiliki konsumen yang berjumlah 50 sampai 417, dan *planning period* yang bervariasi dari 2 sampai 10 hari. PAR-HEM tersebut dibandingkan dengan beberapa *heuristics*, yaitu : CGL – *Metaheuristic* yang diusulkan oleh Cordeau, Gendreau e Laporte; CGW – *Heuristic* yang diusulkan oleh Golden, Chao, e Wasil; dan HGA – *Metaheuristics* yang diusulkan oleh Rocha, Ochi, dan Glover. Dari hasil perbandingan tersebut, dapat dilihat keuntungan dari penggunakan PAR-HEM tersebut. Keuntungan tersebut tidak hanya dari segi waktu *running*, tetapi juga berkaitan dengan kualitas pencarian.

2.2.5. *Dynamic Vehicle Routing Problems* (DVRP)

Semua permasalahan VRP yang telah dibahas sebelumnya, menggunakan data *demand* tiap konsumen yang sudah pasti dan diketahui sejak awal. Jadi, kendaraan hanya perlu ditugaskan untuk memenuhi semua *demand* yang sudah pasti tersebut. Tetapi, pada dunia nyata, ada juga permasalahan VRP dimana semua *demand* tidak diketahui dari awal. Ada sebagian *demand* yang sudah diketahui sebelum rute kendaraan dibuat, tetapi ada juga *demand* yang baru diketahui setelah rute selesai dibuat atau kendaraan sudah berangkat. Jadi, *demand* yang baru tersebut harus dijadwalkan ke rute yang sebelumnya telah dibuat. Permasalahan VRP seperti ini sering disebut dengan *Dynamic Vehicle Routing Problems* (DVRP).

Untuk menyelesaikan pemasalahan DVRP ini, R. Montemanni, L.M.Gambardella, A.E. Rizzoli, dan A.V. Donati memperkenalkan suatu metode yang berdasarkan pada *Ant Colony System*. Permasalahan yang dibahas oleh R. Montemanni, L.M.Gambardella, A.E. Rizzoli, dan A.V. Donati, adalah permasalahan dimana kendaraan diberi tugas yang baru saat sedang dalam perjalanan, dan kendaraan tersebut tidak perlu kembali ke *depot* terlebih dahulu. Model yang diperkenalkan oleh keempat peneliti ini mencakup tiga kelompok aplikasi DVRP, yaitu :

a. *Feeder system*, yang merupakan suatu sistem *dial-a-ride* lokal.

b. *Courier service problem*, contohnya *Federal Express*, dimana *parcel* dikumpulkan dari semua konsumen dan dibawa ke *depot* untuk diproses lebih lanjut.

   Pada kedua aplikasi diatas, saat meninggalkan *depot*, kendaraan dalam keadaan kosong, dan mengumpulkan barang dari para konsumen.

c. Pada aplikasi ketiga ini, kendaraan berangkat dari *depot* dalam keadaan penuh, dan mengirimkannya ke tiap konsumen.

Algoritma yang diusulkan oleh R. Montemanni, L.M.Gambardella, A.E. Rizzoli, dan A.V. Donati, berdasar kepada tiga elemen utama. Pertama adalah *event manager*, yang mengumpulkan order baru, dengan tetap melacak order yang telah dilayani, dan posisi dari tiap – tiap kendaraan pada saat itu. *Event manager* menggunakan informasi ini untuk membangun sebuah urutan dari *static VRP*, yang diselesaikan secara heuristik dengan menggunakan ACS (*Ant Colony System*) *algorithm*, elemen kedua dari algoritma yang diusulkan ini. Elemen ketiga adalah *pheromone conservation procedures*, yang berkaitan erat dengan ACS *algorithm*. Prosedur tersebut digunakan untuk menyampaikan informasi tentang karakteristik dari solusi baik, dari sebuah *static* VRP ke yang selanjutnya.

Dari hasil perhitungan yang telah dilakukan untuk menguji *performance* dari algoritma yang diusulkan ini, dapat dipastikan bahwa performa dari algoritma sangat terkait dengan strategi yang dibangun untuk menyalurkan informasi tentang solusi yang baik. Dari hasil perhitungan lain yang juga telah dilakukan, dapat disimpulkan bahwa algoritma yang diusulkan ini sangat efisien, juga saat dibandingkan dengan teknik heuristik lainnya.

2.2.6. *Multi-Depot Vehicle Routing Problem* (MDVRP)

Pada permasalahan VRP yang dibahas sebelumnya, hanya terdapat satu *depot* saja. Pada permasalahan MDVRP ini, terdapat lebih dari satu *depot*, dimana tiap *depot* memiliki sekelompok kendaraan. Tiap konsumen hanya dapat dilayani oleh satu kendaraan milik *depot* tertentu saja. Tiap kendaraan milik *depot* tertentu, harus mengawali dan mengakhiri perjalanannya di *depot* itu.

Untuk menyelesaikan permasalahan MDVRP, pada umumnya dilakukan dengan dua langkah. Langkah pertama adalah melakukan *cluster*, kemudian baru membuat rute. Pada langkah pertama, permasalahan *clustering* biasanya diselesaikan dengan menggunakan *assignments algorithms*. Total biaya dari solusi untuk permasalahan MDVRP sangat tergantung pada *assignment algorithm* yang digunakan pada langkah pertama, dan algoritma ini sangat tergantung pada *geographic topology* dari permasalahan yang akan diselesaikan. Libertad Tansini, Maria Urquhart, dan Omar Viera, melakukan penelitian untuk membandingkan *assignment algorithm* yang digunakan untuk MDVRP.

Terdapat empat macam *assignment algorithm* yang dibahas oleh Libertad Tansini, Maria Urquhart, dan Omar Viera, yaitu :

a. *Assignment through urgencies*

Algorithm ini menjadwalkan konsumen dengan tingkat *urgency* yang tertinggi terlebih dahulu. *Urgency* merupakan suatu jalan untuk mendefinisikan *precedence relationship* antar konsumen. *Precedence relationship* ini menentukan urutan dari konsumen yang dijadwalkan ke *depot.* Heuristik yang termasuk kelas ini adalah : *Parallel assignment algorithm, Simplified assignment algorithm*, dan *Sweep assignment algorithm.*

b. *Cycling assignment*

Prosedur dari algoritma ini dengan menjadwalkan konsumen satu per satu ke tiap *depot.* Pertama, algoritma menjadwalkan pada tiap *depot*, konsumen yang berjarak paling dekat dengan *depot* tersebut. Kemudian, pada tiap *depot* dijadwalkan konsumen selanjutnya, yang paling dekat dengan *depot* tersebut, sampai semua konsumen telah dijadwalkan pada semua *depot* yang ada.

c. *Assignment by cluster*

Algoritma kelas ini membentuk *compact clusters of customers* untuk tiap *depot.* Sebuah *cluster* didefinisikan sebagai kelompok yang dibangun oleh sebuah *depot*, dan didalamnya dijadwalkan konsumen – konsumen. Saat seorang konsumen dijadwalkan pada suatu *cluster*, maka ini berarti bahwa konsumen ini dijadwalkan untuk *cluster depot* tersebut. Algoritma yang

termasuk pada kelas ini antara lain : *Coefficient Propagation*, dan *Three Criteria Clusterization algorithm*.

d. *Assignment using Transport Problem*

Transport Problem (TP) digunakan untuk menentukan berapa banyak pengiriman yang harus ada antara *depot* dan kosumen. Tiap konsumen harus dilayani minimal oleh satu *depot*. Tujuan dari TP ini adalah meminimumkan total biaya transportasi.

Tujuan dari penelitian yang dilakukan oleh Libertad Tansini, Maria Urquhart, dan Omar Viera ini, adalah untuk membandingkan *assignment algorithm* yang berbeda – beda, dari segi waktu perhitungannya, dan hasil rute untuk permasalahan *original* MDVRP. Hasil dari perbandingan *heuristic algorithm* tersebut menunjukkan tiga kelas algoritma, yaitu : yang memberikan hasil yang baik (dengan waktu perhitungan yang tinggi); waktu perhitungan yang rendah (dengan hasil yang buruk); atau waktu perhitungan medium, dan hasil yang tidak terlalu buruk. Yang termasuk kelas terakhir adalah *urgency algorithms*, sehingga algoritma ini sering dipakail untuk permasalahan nyata yang besar. Juga telah dibuktikan bahwa menggunakan *Transport Problem* sebagai *assignment procedure*, memberikan hasil yang lebih baik daripada *urgency algorithm*. TP dapat menghasilkan hasil yang baik dalam waktu perhitungan yang singkat.

2.2.7. *Multi-Product Vehicle Routing Problem* (MPVRP)

Semua permasalahan VRP yang sudah dibahas sebelumnya, hanya untuk masalah pendistribusian satu macam produk saja. Padahal, ada kalanya perlu didistribusikan beberapa macam produk sekaligus. Oleh karena itu, muncul permasalahan MPVRP, yang membahas mengenai pendistribusian beberapa macam produk sekaligus.

Tujuan dan kendala dari permasalahan MPVRP ini sama dengan tujuan dan kendala dari permasalahan VRP, yaitu untuk meminimumkan total biaya *travel* semua kendaraan, yang harus melayani sejumlah konsumen, dan tiap konsumen hanya dapat dilayani satu kali saja oleh satu kendaraan, dengan memperhatikan kapasitas dari kendaraan. Untuk permasalahan MPVRP ini, ada

sedikit perbedaan pada kendala mengenai kapasitas kendaraan. Pada MPVRP, terdapat sejumlah *P* produk, sehingga jumlah *demand* produk tertentu yang diangkut oleh kendaraan, tidak boleh melebihi kapasitas dari kendaraan tersebut untuk produk itu.

2.2.8. *Vehicle Routing Problem with Pickup and Delivery* (VRPPD)

Guy Desaulniers, Jacques Desrosiers, Andreas Erdmann, Marius M. Solomon, dan Franqois Soumis, membahas mengenai *VRP with Pickup and Delivery* (VRPPD). Pada VRPPD, sejumlah kendaraan *heterogeneous* yang berada pada beberapa *depot*, harus memenuhi sekumpulan permintaan transportasi. Tiap permintaan tersebut terdiri dari sebuah *pickup point*, *delivery point* yang berkorespondensi, dan sebuah *demand* yang harus dikirimkan diantara kedua lokasi tersebut. Ketentuan yang ada dalam VRPPD ini adalah :

- Setiap *pickup* dan *delivery point* hanya dapat dikunjungi tepat satu kali saja.
- Total jumlah *demand* yang diangkut oleh tiap kendaraan tidak boleh melebihi kapasitas dari kendaraan tersebut.
- Untuk tiap pasangan *pickup* dan *delivery point*-nya, keduanya harus berada dalam rute satu kendaraan yang sama, dan *pickup point* tersebut harus didatangi lebih dulu sebelum *delivery point*-nya.
- Untuk tiap rute kendaraan, selalu diawali dan diakhiri di *depot* yang sama.

Apabila ada tambahan kendala *time windows* maka permasalahan VRPPD tersebut menjadi permasalahan *VRPPD with Time Windows* (VRPPDTW). Untuk permasalahan VRPPDTW ini, selain keempat ketentuan diatas, terdapat ketentuan lain, yaitu bahwa tiap kendaran hanya dapat mengunjungi tiap *pickup* dan *delivery points* pada batas *time windows* mereka masing – masing.

Tujuan dari VRPPDTW ini juga sama seperti permasalahan VRP lainnya, yaitu meminimumkan total jarak yang harus ditempuh oleh kendaraan atau total biaya *travel* yang harus dikeluarkan untuk memenuhi semua ketentuan yang ada. Untuk memodelkan permasalahan VRPPDTW ini, Guy Desaulniers,

Jacques Desrosiers, Andreas Erdmann, Marius M. Solomon, dan Franqois Soumis, menggunakan parameter dan variabel sebagai berikut :

- P : *pickup nodes*
- $P = \{1, ..., n\}$
- L : *delivery nodes*
- $L = \{n+1, ..., 2n\}$
- $N = P \cup L$
- d_i = jumlah *demand* pada permintaan i
- $O(k)$: *depot* awal kendaraan k
- $D(k)$: *depot* akhir kendaraan k
- Untuk tiap kendaraan didefinisikan suatu *network* $G_k = (V_k, A_k)$.
- $V_k = N_k \cup \{ O(k), D(k) \}$
- A_k : merupakan *sub set* dari V_k x V_k, yang memuat semua *feasible arcs.*
- $x_{ijk} = \begin{cases} 1, & \text{jika } arc\ (i,j) \in A_k \text{ digunakan oleh kendaraan } k \\ 0, & \text{jika tidak} \end{cases}$
- T_{ik} : variabel waktu saat kendaraan k memulai *service* di *node* $i \in V_k$.
- L_{ik} : jumlah *demand* yang dibawa oleh kendaraan k setelah *service* di *node* $i \in V_k$ selesai dilakukan.

Jika pada permintaan i disebutkan bahwa sejumlah d_i unit harus dikirimkan dari i ke $n+1$, maka $l_i = d_i$, dan $l_{n+i} = - d_i$. Karena tidak setiap kendaraan dapat melayani semua permintaan, maka tiap kendaraan k memiliki satu set $N_k = P_k \cup L_k$ tertentu yang berasosiasi dengan kendaraan tersebut. Tiap kendaraan k diasumsikan berangkat dari *depot* asalnya, $O(k)$, pada waktu $a_{O(k)} = b_{O(k)}$.

Model matematika dari permasalahan VRPPDTW, yang dibuat oleh Guy Desaulniers, Jacques Desrosiers, Andreas Erdmann, Marius M. Solomon, dan Franqois Soumis, adalah :

- Fungsi tujuan :

$$\min \sum_{k \in K} \sum_{(i,j) \in A_k} c_{ijk} x_{ijk} \qquad (2.33)$$

- Kendala :

$$\sum_{k \in K} \sum_{j \in N_k \cup \{D(k)\}} x_{ijk} = 1 \qquad \forall i \in P \qquad (2.34)$$

$$\sum_{j\in N_k} x_{ijk} - \sum_{j\in N_k} x_{j,n+i,k} = 0 \qquad \forall k \in K,\ i \in P_k \tag{2.35}$$

$$\sum_{j\in P_k \cup \{D(k)\}} x_{O(k),j,k} = 1 \qquad \forall k \in K \tag{2.36}$$

$$\sum_{i\in N_k \cup \{O(k)\}} x_{i,j,k} - \sum_{i\in N_k \cup \{D(k)\}} x_{i,j,k} = 0 \qquad \forall k \in K, j \in N_k \tag{2.37}$$

$$\sum_{i\in D_k \cup \{O(k)\}} x_{i,D(k),k} = 1 \qquad \forall k \in K \tag{2.38}$$

$$x_{ijk}(T_{ik} + s_i + t_{ijk} - T_{jk}) \le 0 \qquad \forall k \in K, (i,j) \in A_k \tag{2.39}$$

$$a_i \le T_{ik} \le b_i \qquad \forall k \in K,\ i \in V_k \tag{2.40}$$

$$T_{ik} + t_{i,n+i,k} \le T_{n+i,k} \qquad \forall k \in K,\ i \in P_k \tag{2.41}$$

$$x_{ijk}(L_{ik} + l_j - L_{jk}) = 0 \qquad \forall k \in K, (i,j) \in A_k \tag{2.42}$$

$$l_i \le L_{ik} \le C_k \qquad \forall k \in K,\ i \in P_k \tag{2.43}$$

$$0 \le L_{n+i,k} \le C_k - l_i \qquad \forall k \in K,\ n+i \in D_k \tag{2.44}$$

$$L_{O(k),k} = 0 \qquad \forall k \in K \tag{2.45}$$

$$x_{ijk} \ \ binary \qquad \forall k \in K, (i,j) \in A_k \tag{2.46}$$

Fungsi tujuan (2.33) menunjukkan bahwa tujuan dari VRPPDTW ini adalah meminimumkan total biaya *travel.* Kendala (2.34) dan (2.35) menunjukkan bahwa tiap permintaan (*pickup* dan *delivery nodes*) dilayani oleh tepat satu kali oleh kendaraan yang sama. Untuk memastikan bahwa tiap kendaraan *k* memulai rutenya dari *origin depot O(k)* dan mangakhirinya di *depot D(k)*, maka dibuatlah kendala (2.36) – (2.38). Untuk kendala *time windows*, dibahas pada kendala (2.39) dan (2.40). Untuk tiap permintaan, agar kendaraan mengunjungi *pickup node* sebelum *delivery node*, maka dibuat kendala (2.41). Kendala (2.42) menunjukkan compatibility yang diperlukan, antara rute, dan *loads* kendaraan. Kendala (2.43), dan (2.44), menunjukkan interval batas kapasitas kendaraan di *pickup* dan *delivery nodes*. Kendala (2.45) menunjukkan bahwa *initial load* untuk tiap kendaraan adalah 0.

Mikkel Sigurd, David Pisinger, dan Michael Sig, membahas mengenai permasalahan *Pickup and Delivery Problem with Time Windows and Precedences* (PDPTWP). PDPTWP merupakan salah satu variasi dari permasalahan VRP,

dimana kendaraan harus mengirimkan beberapa barang, antar kelompok konsumen. Tiap konsumen memiliki *time windows, precedence number,* dan suatu *quantity*. Tiap kendaraan harus mengunjungi konsumen dalam batas *time windows* mereka, dalam urutan *precedence* yang *nonincreasing*, dan kapasitas dari kendaraan juga harus diperhatikan. Kendaraan dilokasikan pada beberapa *depot,* dan tiap kendaraan memiliki kapasitas yang tidak boleh dilampaui. Diasumsikan bahwa kendaraan meninggalkan *depot* tanpa membawa kargo, dan kembali ke *depot* yang sama, juga tanpa membawa kargo.

Tujuan dari permasalahan PDPTWP ini adalah untuk memenuhi sejumlah kiriman dengan biaya pengiriman yang paling rendah. Tiap pengiriman terdiri dari seorang penjual dan seorang pembeli. Setelah kendaraan mengambil barang dari penjual, maka kendaraan tersebut harus langsung mengirimkannya ke pembeli yang bersangkutan. Ada juga situasi dimana suatu pengiriman terdiri dari lebih dari satu penjual dengan satu pembeli yang sama, atau lebih dari satu pembeli, dengan satu penjual yang sama. Pada situasi seperti itu, kendaraan diperbolehkan untuk pertama – tama mengambil semua kargo dari semua penjual yang ada, kemudian baru mengirimkannya ke pembeli. Pada PDPTWP, kendaraan harus mengunjungi konsumen sesuai dengan urutan *precedence* secara *nonincreasing*.

Mikkel Sigurd, David Pisinger, dan Michael Sig, menggambarkan permasalahan PDPTWP sebagai suatu *oriented graph* $G = (N, E)$, dimana *node set* N terdiri dari konsumen C dan *depot* D. Tiap konsumen memiliki suatu *quantity* q_i. q_i yang bernilai positif mempunyai arti bahwa kendaraan harus mengambil sejumlah barang dari konsumen i, sehingga konsumen i tersebut dapat disebut sebagai penjual. Konsumen yang mempunyai nilai q_i negatif adalah pembeli. Pada tiap konsumen $i \in C$, diberikan suatu *time window* $[a_i, b_i]$, dan suatu kode *precedence* p_i. Kendaraan harus mengunjungi konsumen dalam batas *time window* mereka, dan sesuai dengan nomer *precedence* yang *nonincreasing*.

Konsumen dilayani oleh sekumpulan kendaraan V. Tiap kendaraan $k \in V$, memiliki kapasitas Q_k, suatu *start depot* D_k^+, dan suatu *end depot* D_k^-. Sekumpulan *start depot* dinyatakan sebagai $D^+ = U_{k \in V} D_k^+$, sedangkan

sekumpulan *end depot* dinyatakan sebagai $D^- = U_{k \in V} D_k^-$. Tiap *depot* $d \in D = D^+ \cup D^-$, memiliki *time window* [a_d, b_d]. Biaya *travel* dari *node i* ke *node j* dinyatakan dengan c_{ij}, dan waktu *travel*-nya diasumsikan $t_{ij} > 0$. Jika tidak ada *edge* antara kedua *node i* dan *j*, maka biaya c_{ij} dapat dibuat tak terhingga.

Variabel yang digunakan pada model permasalahan PDPTWP ini, adalah:

- $x_{ijk} = \begin{cases} 1, & \text{jika kendaraan } k \text{ dari } node\ i \text{ langsung ke } node\ j \\ 0, & \text{jika tidak} \end{cases}$
- s_i : waktu kendaraan sampai di *node i* (yang dapat berupa konsumen atau *depot*).
- y_{ik} : kapasitas dari kendaraan k saat sampai di konsumen *i.*

Jika kendaraan k tidak mengunjungi *node i*, maka $s_{ik} = 0$ dan $y_{ik} = 0$.

Model dari permasalahan PDPTWP yang diperkenalkan oleh Mikkel Sigurd, David Pisinger, dan Michael Sig, adalah :

- Fungsi tujuan :

$$\text{Min} \sum_{k \in V} \sum_{i \in N} \sum_{j \in N} c_{ij} x_{ijk} \tag{2.47}$$

- Kendala :

$$\sum_{k \in V} \sum_{j \in N} x_{ijk} = 1, \qquad \forall i \in C \tag{2.48}$$

$$\sum_{j \in N} x_{D_k^+ jk} = 1, \qquad \forall k \in V \tag{2.49}$$

$$\sum_{i \in N} x_{iD_k^- k} = 1, \qquad \forall k \in V \tag{2.50}$$

$$\sum_{i \in N} x_{ihk} - \sum_{j \in N} x_{hjk} = 0, \qquad \forall h \in C, \ \forall k \in V \tag{2.51}$$

$$y_{D_k^- k} = 0, \qquad \forall k \in V \tag{2.52}$$

$$\sum_{i \in N} Q_k x_{ijk} \geq y_{jk}, \qquad \forall j \in N, \ \forall k \in V \tag{2.53}$$

$$y_{ik} + x_{ijk} q_i - K(1 - x_{ijk}) \leq y_{jk}, \qquad \forall i \in N, \ \forall j \in N, \ \forall k \in V \tag{2.54}$$

$$s_{ik} + t_{ij} - K(1 - x_{ijk}) \leq s_{jk}, \qquad \forall i \in N, \ \forall j \in N, \ \forall k \in V \tag{2.55}$$

$$p_i - K(1 - x_{ijk}) \leq p_j, \qquad \forall i \in C, \ \forall j \in C, \ \forall \text{k} \in V \tag{2.56}$$

$$a_i \le s_{ik} \le b_i, \qquad \forall i \in N, \ \forall k \in V \qquad (2.57)$$

$$y_{ik} \ge 0, \qquad \forall i \in N, \ \forall k \in V \qquad (2.58)$$

$$s_{ik} \ge 0, \qquad \forall i \in N, \ \forall k \in V \qquad (2.59)$$

$$x_{ijk} \in \{0,1\}, \qquad \forall i \in N, \ \forall j \in N, \ \forall k \in V \qquad (2.60)$$

Kendala (2.48) memastikan bahwa tiap konsumen hanya dikunjungi tepat satu kali saja. Kendala (2.49), dan (2.50) memastikan bahwa tiap kendaraan berangkat dari *start depot*-nya dan kembali ke *end depot*-nya. Kendala (2.51) memastikan bahwa jika suatu kendaraan datang ke suatu *node*, maka kendaraan tersebut juga harus berangkat dari *node* tersebut. Karena waktu kedatangan, s_{ik}, dari kendaraan akan selalu meningkat, maka tidak akan ada *subtour* dalam solusinya, sehingga kendala (2.49) – (2.51) menjamin bahwa tiap kendaraan akan mengikuti suatu *path* dari *start depot* ke *end depot*-nya. Kapasitas dari kendaraan dibatasi oleh kendala (2.52) – (2.54), sedangkan kendala *time window* terdapat pada kendala (2.55) dan (2.56). Kendala (2.57) memastikan bahwa tiap *node* dikunjungi sesuai dengan *precedency*. Konstanta K adalah bilangan yang sangat besar.

2.2.9. *Split Delivery Vehicle Routing Problem* (SDVRP)

C. Archetti, dan M. G. Speranza (2001), membahas mengenai k-*Split Delivery Vehicle Routing Problem* (k-SDVRP). Pada permasalahan k-SDVRP, terdapat sekumpulan kendaraan *homogeneous* yang harus melayani sekelompok konsumen, yang memiliki *demand*, yang kemungkinan lebih besar daripada kapasitas kendaraan yang dinyatakan dengan $k, k \in Z^+$. Berbeda dengan asumsi yang digunakan pada *Capacitated Vehicle Routing Problem* (CVRP), pada SDVRP ini, tiap konsumen boleh dilayani lebih dari satu kali, walaupun permintaan dari suatu konsumen tidak melebihi kapasitas maksimum dari kendaraan. Setiap kali kendaraan melayani seorang konsumen, maka kendaraan tersebut akan mengirimkan suatu *integer quantity*. Jumlah kendaraan yang tersedia tidak terbatas. Terdapat sebuah *depot*, dan tiap kendaraan harus mengawali dan mengakhiri *tour*-nya di *depot* tersebut. Tujuan dari permasalahan

SDVRP ini adalah untuk menemukan sekumpulan rute kendaraan yang melayani semua konsumen, dimana jumlah *quantity* yang dikirimkan pada tiap *tour* tidak melebihi kapasitas dari kendaraan, dan total jarak yang ditempuh kendaraan adalah minimum. k-SDVRP merupakan permasalahan SDVRP (yang merupakan permasalahan VRP, dimana tiap konsumen dapat dikunjungi lebih dari satu kali), dimana *demand* dari tiap konsumen, dan juga *quantity* yang dikirimkan oleh kendaraan saat mengunjungi konsumen, adalah sebuah *integer number*.

Chi-Guhn Lee, Marina Epelman, Chelsea C. White III, dan Yavuzm A. Bozer (2002), membahas mengenai *Multiple-Vehicle Routing Problem with Split Pick-ups* (mVRPSP). Permasalahan yang dibahas oleh keempat peneliti tersebut, berhubungan dengan sekumpulan kendaraan yang memiliki kapasitas yang identik, *multiple suppliers*, dan *single assembly plant*. Kendaraan bertanggung jawab untuk memindahkan *parts* dari *suppliers* ke *assembly plant*. Permasalahannya adalah untuk menentukan berapa banyak *parts* yang harus diambil dari masing – masing *supplier*, oleh tiap – tiap kendaraan, yang akan dikirimkan ke *assembly plant*. Tujuannya adalah untuk meminimumkan total biaya transportasi kendaraan. Biaya ini tergantung pada jumlah kendaraan yang digunakan, dan rutenya, tetapi tidak tergantung pada jumlah *parts* yang ditransportasikan. Pada permasalahan mVRPSP, tiap *suppliers* dapat dikunjungi lebih dari satu kali.

Pada permasalahan mVRPSP ini ada dua hal yang perlu dilakukan, yaitu:

- Menentukan jumlah kendaraan yang diperlukan untuk mengirimkan *parts* dari sekumpulan *suppliers* ke *single assembly plant.*
- Menentukan *suppliers* yang harus dikunjungi oleh tiap kendaraan, dan jumlah *parts* yang harus diambil pada masing – masing *suppliers*, dengan tujuan meminimumkan total biaya transportasi.

Untuk memodelkan permasalahan mVRPSP, Chi-Guhn Lee, Marina Epelman, Chelsea C. White III, dan Yavuzm A. Bozer (2002), menggunakan parameter dan variabel sebagai berikut :

- M : jumlah *suppliers*.
- $M = 0$ adalah *depot* atau *assembly plant*.

- $a_i \geq 0$, $i = 1, ..., M$, adalah jumlah *supply* pada *supplier* i, atau dapat disebut sebagai jumlah *truckloads*.
- U : *upper bound* dari jumlah kendaraan yang dibutuhkan untuk memindahkan semua *supply*.
- $k = 1, ..., U$
- $i, j = 0, ..., M$
- $x_{ijk} = \begin{cases} 1, & \text{jika kendaraan } k \text{ dari supplier } i \text{ langsung ke supplier } j \\ 0, & \text{jika tidak} \end{cases}$
- $y_{ik} = \begin{cases} 1, & \text{jika } \textit{supplier } i \text{ dilayani oleh kendaraan } k \\ 0, & \text{jika tidak} \end{cases}$
- w_{ik} : jumlah *load* yang diambil dari *supplier* i oleh kendaraan k.
- u_{ik} : *dummy continuous variables* untuk kendala penghilangan *subtour.*

Chi-Guhn Lee, Marina Epelman, Chelsea C. White III, dan Yavuzm A. Bozer (2002), memodelkan permasalahan mVRPSP dengan menggunakan *Mixed integer programming* (MIP). MIP *formulation* tersebut adalah sebagai berikut :

- Fungsi tujuan :

$$\min \sum_i \sum_j c_{ij} \sum_k x_{ijk} \tag{2.61}$$

- Kendala :

$$\sum_j x_{ijk} = \sum_j x_{jik} = y_{ik} \; ; \qquad \forall i, \forall k \tag{2.62}$$

$$u_{ik} - u_{jk} + (M+1).x_{ijk} \leq M \; ; \qquad \forall i \; (\neq 0), \forall j \; (\neq 0), \forall k \tag{2.63}$$

$$w_{ik} \leq a_i y_{ik} \; ; \qquad \forall i, \forall k \tag{2.64}$$

$$\sum_k w_{ik} = a_i \; ; \qquad \forall i \tag{2.65}$$

$$\sum_i w_{ik} \leq 1 \; ; \qquad \forall k \tag{2.66}$$

$$y_{ik}, x_{ijk} \in \{0,1\} \; ; \qquad \forall i, \forall j, \forall k \tag{2.67}$$

$$w_{ik} \geq 0 \; ; \qquad \forall i, \forall k \tag{2.68}$$

Fungsi kendala (2.61) menunjukkan total biaya *travel* sebagai jumlah dari biaya *travelling* diantara *consecutive suppliers*. Kendala (2.62) merupakan

kendala *flow conservation*, dan digunakan sebagai definisi dari variabel y_{ik}. Kendala (2.63) merupakan kendala penghilangan *subtour* untuk rute dari tiap – tiap kendaraan. Kendala (2.64) memastikan bahwa *pickups* hanya dilakukan dengan melakukan kunjungan ke *suppliers*. Kendala (2.65) menjamin bahwa semua *supplies* diambil. Kendala (2.66) memastikan bahwa total *load* untuk tiap kendaraan tidak melebihi kapasitas dari kendaraan.

2.2.10. *Inventory Routing Problem* (IRP)

Ann Campbell, Lloyd Clarke, Anton Kleyweg, dan Martin Savelsberg (1997), membahas mengenai *Inventory Routing Problem* (IRP). IRP muncul sebagai akibat dari adanya *Vendor Managed Inventory Replenishment* (VMI), yang pada akhir – akhir ini telah digunakan oleh banyak perusahaan.Yang dimaksud dengan VMI adalah situasi dimana *vendor* (penjual) memonitor level inventori konsumennya dan menentukan kapan serta berapa banyak inventori yang dibutuhkan oleh tiap – tiap konsumen.

Permasalahan IRP yang dibahas oleh Ann Campbell, Lloyd Clarke, Anton Kleyweg, dan Martin Savelsberg (1997), merupakan permasalahan yang berhubungan distribusi suatu produk tertentu yang berulang, dari suatu fasilitas tertentu, ke sekumpulan *n* konsumen, dalam suatu batas *planning period T*, yang kemungkinan tidak terbatas. Konsumen *i* mengkonsumsi produk tersebut dengan tingkat kecepatan rata – rata, u_i, (volume per hari), dan memiliki kemampuan untuk mempertahankan inventori lokal dari produk tersebut sampai suatu batas maksimum C_i. Inventori dari konsumen *i*, pada waktu *0* adalah I_i. Tersedia sejumlah kendaraan *homogeneous*, dengan kapasitas *Q*, yang akan mendistribusikan produk tersebut. Tujuan dari IRP ini adalah untuk meminimumkan rata – rata biaya distribusi selama *planning period*, tanpa menyebabkan terjadinya *stockout* pada tiap konsumen.

Untuk permasalahan IRP ini, ada tiga keputusan yang harus diambil, yaitu :

- Kapan waktu melayani suatu konsumen ?
- Berapa jumlah produk yang harus dikirimkan ke konsumen tersebut ?

- Rute pengiriman mana yang akan digunakan ?

Permasalahan IRP ini berbeda dari permasalahan VRP lainnya, karena pada IRP ini lebih berdasar kepada tingkat penggunaan dari konsumen, dan bukannya permintaan atau *order* dari konsumen tersebut.

Permasalahan IRP yang dibahas diatas, termasuk permasalahan yang *deterministic* dan *static*, sebab diasumsikan bahwa tingkat penggunaan konsumen diketahui dan tetap. Pada kenyataan di dunia nyata, permasalahan ini adalah *stochastic* dan *dynamic*. Oleh karena itu, muncul variasi dari IRP yang disebut *Stochastic Inventory Routing Problem* (SIRP). Pada permasalahan SIRP, permintaan atau kebutuhan dari konsumen di masa mendatang tidak menentu.

Hoong Chuin Lau, dan Qi Zhang Liu (1999), mambahas mengenai *Inventory Routing Problem with Time Windows* (IRPTW). Permasalahan yang dibahas oleh kedua peneliti tersebut, berhubungan dengan suatu sistem distribusi, yang mempunyai *multiple suppliers*, *capacitated warehouses*, *capacitated retailers*, *identical capacitated* kendaraan, dan *unit-sized items*. *Items* tersebut harus dikirimkan dari *suppliers* ke *warehouse*, kemudian dikirimkan ke *retailers*, oleh kendaraan. Kendaraan dapat melakukan pengiriman ke *multiple retailers*, dalam batas *time windows* dari masing – masing *retailers*. *Retailers' time-varying demand forecast* pada suatu *planning horizon*, diketahui. Tujuan dari permasalahan ini adalah untuk mencari suatu rencana distribusi, yang meminimumkan total biaya operasi, yang termasuk biaya *inventory* (untuk jumlah yang melebihi *demand*), biaya *backlogging* (untuk jumlah yang kurang dari *demand*), dan biaya transportasi. Permasalahan IRPTW merupakan permasalahan yang kompleks, sebab IRPTW merupakan integrasi dari dua permasalahan optimasi klasik : *dynamic capacitated lot-sizing problem*, dan VRPTW.

Asumsi yang digunakan oleh Hoong Chuin Lau, dan Qi Zhang Liu (1999), adalah :

- *Single warehouse.*
- Tiap *supplier* memiliki *supply* yang tidak terbatas untuk semua *items*.
- *Initial inventory levels* pada *warehouse* dan *retailers* adalah 0.
- Rute dari *suppliers* ke *warehouse* tidak akan diperhitungkan (walaupun jumlah yang dikirimkan akan dihitung).

Hoong Chuin Lau, dan Qi Zhang Liu (1999), memecah permasalahan IRPTW menjadi dua *sub problem* (distribusi dan *routing*), ditambah suatu mekanisme *interface* yang memperbolehkan kedua algoritma tersebut bekerja sama dalam suatu *master-slave fashion*, dengan algoritma distribusi (A1) mengontrol algoritma *routing* (A2). Prosedurnya adalah sebagai berikut : A1 akan menentukan jumlah *flow* antara *suppliers*, *warehouse*, dan *retailers*, yang meminimumkan biaya *inventory*, dan *backlogging*, yang berkaitan dengan kapasitas dari *warehouse*, dan *retailers*. A2, berdasarkan jumlah *flow* yang diberikan (atau dapat disebut sebagai *demand* konsumen), membentuk rute pengiriman, yang meminimumkan biaya transportasi, yang berkaitan dengan kapasitas kendaraan dan *time windows*.

Parameter dan variabel yang digunakan oleh Hoong Chuin Lau, dan Qi Zhang Liu (1999), untuk memodelkan permasalahan IRPTW adalah :

- S : sekelompok *suppliers*.
- R : sekelompok *retailers*.
- J : sekelompok *items*.
- T : *consecutive days* dalam *planning period* { 1, 2, ..., n }
- D_{ijt} : *demand retailer* i untuk produk j pada hari t.
- Q_v : kapasitas kendaraan.
- Q_W : kapasitas *storage warehouse*.
- Q_i : kapasitas *storage* dari *retailer* i.
- W_i : *time window* dari *retailer* i.
- C_j : *inventory holding cost* per unit *item* j per hari di *warehouse*.
- C_{ij} : *inventory holding cost* per unit *item* j per hari di *retailer* i.
- B_{ij} : biaya *backlogging* per unit *item* j per hari di *retailer* i.
- T_{ik} : biaya transportasi dari *retailer* i ke *retailer* k.
- z_{ijt} : jumlah *integral item* j yang terdapat pada *retailer* i, pada hari t.
- z_{jt} : jumlah *integral item* j yang terdapat pada *warehouse*, pada hari t.
- b_{ijt} : jumlah *integral backlog item* j untuk retailer i, pada hari t.
- x_{sjt} : jumlah *integral flow item* j dari *supplier* s ke *warehouse*, pada hari t.
- x_{ijt} : jumlah *integral flow item* j dari *warehouse* ke *retailer* i, pada hari t.

- c_r : biaya dari rute r, yang merupakan jumlah dari biaya transportasi T_{ik} dari semua *adjacent retailers* i dan k dalam rute r.
- f : sekelompok rute transportasi harian.
- $y_r = \begin{cases} 1, \text{ jika rute } r \in f \text{ digunakan} \\ 0, \text{ jika tidak} \end{cases}$
- $y_{it} = \begin{cases} 1, \text{ jika retailer } i \text{ dilayani pada hari } t \\ 0, \text{ jika tidak} \end{cases}$
- $h_{irt} = \begin{cases} 1, \text{ jika retailer } i \text{ dilayani oleh rute } r \in f, \text{ pada hari } t \\ 0, \text{ jika tidak} \end{cases}$

Model dari permasalahan IRPTW yang dibuat oleh Hoong Chuin Lau, dan Qi Zhang Liu (1999), adalah :

- Fungsi tujuan :

$$\min \sum_j \sum_t \left(z_{jt} C_j + \sum_i z_{ijt} C_{ij} + \sum_i b_{ij} Bij \right) + \sum_r y_r c_r \qquad (2.69)$$

- Kendala :

$$\sum_j z_{jt} \le Q_W \ ; \qquad \text{untuk } t \in T \qquad (2.70)$$

$$\sum_j x_{ijt} + \sum_j z_{ijt} \le Q_i \ ; \qquad \text{untuk } i \in R, \text{ dan } t \in T \qquad (2.71)$$

$$\sum_i x_{ijt} \le z_{jt} \ ; \qquad \text{untuk } j \in J, \text{ dan } t \in T \qquad (2.72)$$

$$\sum_s x_{sjt} + z_{jt} - z_{j,t+1} - \sum_i x_{ijt} = 0 \ ; \qquad \text{untuk } j \in J, \text{ dan } t < n \qquad (2.73)$$

$$x_{ijt} + z_{ijt} - z_{j,t+1} - b_{ijt} + b_{ij,t+1} = D_{ijt} \ ; \text{ untuk } i \in R, j \in J, \text{ dan } t < n \qquad (2.74)$$

$$x_{ijt} + z_{ijt} - b_{ijt} = D_{ijt} \ ; \qquad \text{untuk } i \in R, j \in J, \text{ dan } t = n \qquad (2.75)$$

$$\sum_j x_{ijt} \le y_{it} Q_i \ ; \qquad \text{untuk } i \in R, \text{ dan } t \in T \qquad (2.76)$$

$$y_{it} \le y_r \ ; \qquad \text{untuk } i \in R, r \in f, t \in T, \text{dengan } h_{irt} = 1 \qquad (2.77)$$

$$\sum_i \sum_j \sum_t x_{ijt} h_{irt} \le y_r Q_V ; \qquad \text{untuk } r \in f \qquad (2.78)$$

Fungsi tujuan (2.69), terdiri dari empat komponen, yaitu : biaya *inventory warehouse* (C_j), biaya *inventory retailers* (C_{ij}), biaya *backlogging* (B_{ij}), dan

biaya transportasi dari *warehouse* ke *retailers* (T_{ik}) . Kendala (2.70) adalah kendala dari kapasitas *warehouse,* dan kendala (2.71) adalah kendala kapasitas *retailers*. Kendala (2.72) berarti bahwa inventori dalam *warehouse* harus melebihi kebutuhan pengiriman tiap hari. Kendala (2.73) merupakan kendala *inventory balance* pada *warehouse,* sedangkan kendala (2.74) dan (2.75) merupakan kendala *inventory balance* pada *retailers*. Kendala (2.76) digunakan untuk memastikan apakah tiap *retailer* dilayani pada tiap harinya. Kendala (2.77) berarti bahwa sebuah rute digunakan jika terdapat paling tidak satu *retailer* dalam rute tersebut yang dilayani. Kendala (2.78) merupakan kendala dari kapasitas kendaraan.