ISSN 2339-1499

Paper ini telah direview dan dipublikasikan di Jurnal Rekayasa Sistem Industri
Volume 6 No.1 April 2017
http://journal.unpar.ac.id/index.php/jrsi/index
ISSN: 0216-1036 (print) & ISSN 2339-1499 (online)

Pengembangan Algoritma *Hybrid Restart Simulated Annealing with Variable Neighborhood Search* (HRSA-VNS) untuk penyelesaian kasus *Vehicle Routing Problem with Time Windows* (VRPTW)

Titi Iswari[1]
[1)] Fakultas Teknologi Industri, Jurusan Teknik Industri, Universitas Katolik Parahyangan
Jl. Ciumbuleuit 94, Bandung 40141
Email: titiiswari@unpar.ac.id

Abstract

Determining the vehicle routing is one of the important components in existing logistics systems. It is because the vehicle route problem has some effect on transportation costs and time required in the logistics system. In determining the vehicle routes, there are some restrictions faced, such as the maximum capacity of the vehicle and a time limit in which depot or customer has a limited or spesific opening hours (time windows). This problem referred to Vehicle Routing Problem with Time Windows (VRPTW). To solve the VRPTW, this study developed a meta-heuristic method called Hybrid Restart Simulated Annealing with Variable Neighborhood Search (HRSA-VNS). HRSA-VNS algorithm is a modification of Simulated Annealing algorithm by adding a restart strategy and using the VNS algorithm scheme in the stage of finding neighborhood solutions (neighborhood search phase). Testing the performance of HRSA-VNS algorithm is done by comparing the results of the algorithm to the Best Known Solution (BKS) and the usual SA algorithm without modification. From the results obtained, it is known that the algorithm perform well enough in resolving the VRPTW case with the average differences are -2.0% with BKS from Solomon website, 1.83% with BKS from Alvarenga, and -2.2% with usual SA algorithm without any modifications.

Keywords : vehicle routing problem, time windows, simulated annealing, VNS, restart

Abstrak

Penentuan rute kendaraan (vehicle routing problem) merupakan salah satu komponen penting dalam sistem logistik saat ini. Hal ini dikarenakan penentuan rute kendaraan berpengaruh pada biaya transportasi dan waktu yang diperlukan dalam sistem logistik. Dalam permasalahan penentuan rute kendaraan terdapat batasan yang dihadapi yaitu kapasitas maksimum dari kendaraan dan batasan waktu dimana depot atau customer memiliki jam buka yang terbatas atau spesifik (time windows). Permasalahan inilah yang disebut sebagai Vehicle Routing Problem with Time Windows (VRPTW). Untuk menyelesaikan permasalahan VRPTW, penelitian ini mengembangkan suatu metode metaheuristik yaitu Hybrid Restart Simulated Annealing with Variable Neighborhood Search (HRSA-VNS). Algoritma HRSA-VNS ini merupakan modifikasi dari algoritma Simulated Annealing dengan menambahkan strategi restart dan menggunakan skema algoritma VNS dalam tahap pencarian solusi tetangga (neighborhood search). Pengujian performa algoritma HRSA-VNS dilakukan dengan membandingkan hasil algoritma dengan Best Known Solution (BKS) dan algoritma SA biasa tanpa modifikasi. Dari hasil yang didapatkan, diketahui bahwa algoritma yang dibangun cukup baik dalam menyelesaikan kasus VRPTW dengan rata-rata difference-nya adalah -2,0% dengan BKS web Solomon, 1,83% dengan BKS dari Alvarenga, dan -2,2% dengan algoritma SA biasa tanpa modifikasi.

Kata kunci : penentuan rute, time windows, simulated annealing, VNS, restart

Pendahuluan

Masalah mengenai pendistribusian barang dalam bidang logistik menjadi salah satu masalah yang sering terjadi saat ini. Menurut Xie, et al. (2008) seiring dengan bertumbuhnya ekonomi dari suatu daerah, *demand* atau permintaan pada sistem logistik juga semakin meningkat. Hal ini membuat kapasitas dari transportasi dan sistem pendistribusian juga harus ditingkatkan agar dapat tercipta sistem logistik yang efisien.

Salah satu cara yang efektif dalam peningkatan performansi distribusi dalam sistem logistik adalah dengan melakukan penentuan rute kendaraan yang paling optimal. Penentuan rute kendaraan (*vehicle routing problem*) dalam sistem logistik

merupakan salah satu komponen penting karena berpengaruh pada total biaya transportasi dan waktu operasi pada jaringan logistik tersebut.

Dalam *vehicle routing problem* terdapat satu atau lebih *depot* yang akan melayani sejumlah *customer* dengan adanya batasan tertentu. Batasan yang paling umum adalah adanya kapasitas maksimum dari kendaraan yang digunakan dalam sistem logistik tersebut. Pada kenyataannya selain terdapat batasan kapasitas dari kendaraan yang digunakan seringkali terdapat juga batasan waktu dimana *depot* atau *customer* memiliki jam buka yang terbatas atau spesifik (*time windows*) dalam melakukan proses pengambilan atau pendistribusian barang. Permasalahan inilah yang disebut sebagai *Vehicle Routing Problem with Time Windows (VRPTW)*.

Vehicle Routing Problem with Time Windows merupakan permasalahan *NP-hard* (*Nondeterministic Polynomial Time*) sehingga metode *exact optimization* sulit untuk menyelesaikan kasus VRP. Apabila dapat diselesaikan dengan metode eksak maka akan menghabiskan waktu yang sangat lama terutama untuk jumlah *nodes* yang besar (*large instance*). Untuk mendapatkan solusi yang relevan dengan kondisi real dan sangat dekat dengan solusi yang optimal maka digunakanlah metode heuristik atau metaheuristik (Kumar dan Panneerselvam, 2012).

Penelitian ini mengajukan sebuah metode metaheuristik untuk penyelesaian VRPTW yaitu *Hybrid Restart Simulated Annealing with Variable Neighborhood Search* (HRSA-VNS). Algoritma HRSA-VNS memodifikasi algoritma *Simulated annealing* (SA) dengan menambahkan strategi restart dan menggunakan skema algoritma VNS dalam tahap pencarian solusi tetangga (*neighborhood search*). Penelitian ini mengembangkan algoritma *restart simulated annealing* dari Iswari, et al. (2016). Strategi ini digunakan untuk membuat algoritma SA dapat menjelajahi porsi yang lebih besar dari ruang pencarian dengan harapan dapat menemukan solusi global optimal. Strategi ini dilakukan untuk membuat diversifikasi pencarian dengan menghindari terjebak dalam solusi *local optimal*.

Tinjauan Pustaka

VRP pertama kali diperkenalkan oleh Dantzig and Ramser pada tahun 1959. Pada penelitian tersebut, mereka menentukan rute optimal pada pendistribusian gasolin dari satu terminal besar ke stasiun-stasiun yang disuplai oleh terminal tersebut. VRP dapat dideskripsikan sebagai suatu masalah untuk mendesain rute pengumpulan yang optimal dari satu atau beberapa *depot* ke beberapa *customer* dengan mempertimbangkan batasan-batasan yang ada (Laporte, 1992).

Laporte (1992) mendefinisikan VRP sebagai berikut : Diketahui G (V,A) dengan V = *{1,.....,n}* merepresentasikan kumpulan *customer* dengan *demand* yang yang diketahui dan deterministik. V (1) merepresentasikan adanya satu *depot*. A merupakan kumpulan *arc* yang menghubungkan semua *customer*. Biaya dari setiap *arc* antara *nodei* dan *j* ditunjukkan dengan c_{ij}. Fungsi tujuannya adalah untuk meminimalkan total jarak tempuh dari setiap kendaraan yang ada dengan memperhatikan batasan seperti :

1. Satu *customer* dikunjungi hanya sekali
2. Setiap rute berawal dan berakhir pada *depot*
3. Batasan lain yang harus dipenuhi sesuai kasus yang ada (kapasitas, *time windows, maximum time*, dll)

VRP yang paling umum digunakan adalah *Capacitated Vehicle Routing Problem* (CVRP). Pada CVRP, kendaraan mempunyai kapasitasnya masing-masing dan total produk yang diangkut tidak boleh melebihi kapasitas yang ada (Solomon, 1988). Namun pada kenyataannya, terkadang selain terdapat batasan kapasitas kendaraan, konsumen juga mungkin memiliki suatu periode waktu dimana mereka hanya bisa dilayani pada jam tersebut. Kasus ini disebut *Vehicle Routing Problem with Time Windows (VRPTW)* (Solomon, 1987). Berikut ini adalah model matematis dari VRPTW (Cordeau, et al., 2000)

Notasi

N = Jumlah pelanggan{0,1,2,….,|N|}
K = Jumlah kendaraan {1,2,….,|K|}
t_{ij} = Waktu tempuh dari *node* i ke j
o_i = Waktu buka pada *node* i
e_i = Waktu tutup pada *node* i
C = Kapasitas kendaraan
a_{ik} = Waktu kedatangan kendaraan k pada *node* i
d_{ij} = Jarak dari *node* i ke *node* j
p_j = Demand pada *node* j

Variabel keputusan

x_{ijk} = 1 jika kendaraan k menempuh perjalanan dari *node* i ke*node* j,
0 jika kendaraan k tidak menempuh perjalanan dari *node* i ke *node* j

Minimize

$$\sum_{k\in K}\sum_{i\in N, i\neq j}\sum_{j\in N, j\neq i} d_{ij}x_{ijk} \qquad \text{Pers. 1}$$

Subject to:

$$\sum_{k\in K}\sum_{i\in N'} x_{ijk} = 1 \quad \text{Pers. 2}$$

$$\sum_{i\in N'} x_{0ik} \le 1 \quad \text{Pers. 3}$$

$$\sum_{i\in N, i\neq j} x_{ilk} = \sum_{j\in N, j\neq i} x_{ljk} \quad \text{Pers. 4}$$

$$o_0 < a_{0k} \le e_0 \quad \text{Pers. 5}$$

$$e_i > a_{ik} > o_i \quad \text{Pers. 6}$$

$$a_{jk} \ge a_{ik} + t_{ij} - M(1 - x_{ijk}) \quad \text{Pers. 7}$$

$$\sum_{i\in N, i\neq j}\sum_{j\in N, j\neq i} x_{ijk} t_{ij} \le e_0 \quad \text{Pers. 8}$$

$$\sum_{i\in N, i\neq j}\sum_{j\in N, j\neq i} p_j x_{ijk} \le C \quad \text{Pers. 9}$$

Fungsi tujuan (Pers. 1) adalah untuk meminimalkan total jarak tempuh. Satu *node* hanya dapat dikunjungi sekali oleh satu kendaraan saja didefinisikan oleh Pers 2, dan setiap kendaraan yang digunakan paling banyak satu kali seperti yang didefinisikan oleh Pers 3. Pers 4 adalah *flow constraint* dari kendaraan yang ada. Pers 5, 6, 7, dan 8 merupakan *time windows constraint*. Semua waktu kedatangan dan rute harus berada dalam jendela waktu. Pers 9 menjamin bahwa semua produk yang diambil atau didistribusikan oleh kendaraan harus kurang dari atau sama dengan kapasitas kendaraannya.

VRP pada umumnya diselesaikan dengan menggunakan metode eksak atau metode heuristik/metaheuristik. Penggunaan metode eksak sendiri memiliki kelebihan yaitu akan mendapatkan solusi global optimal tetapi akan menghabiskan waktu yang sangat lama terutama untuk jumlah *nodes* yang besar (*large instance*). Untuk mendapatkan solusi yang relevan dengan kondisi real dan sangat dekat dengan solusi yang optimal maka digunakanlah metode heuristik atau meta-heuristik (Kumar dan Panneerselvam, 2012).

Penelitian ini menggunakan salah satu metode metaheuristik yaitu metode *simulated annealing* yang telah dimodifikasi. Metode *simulated annealing* digunakan sebagai metode dasar karena *Simulated annealing* sudah terbukti menghasilkan performa yang baik dalam menyelesaikan kasus VRP (Alfa et al., 1991; Breedam, 1995; Lin et al., 2006; Kuo, 2010; Lin et al., 2011; Xiao et al., 2012). Selain itu, SA juga telah berhasil menyelesaikan permasalahan-permasalahan *complex combinatorial optimization* (Breedam, 1995; Yu et al., 2010).

Simulated annealing dengan strategi restart juga pernah dikembangkan oleh beberapa peneliti seperti Oliviera, et al. (2006), Yu dan Lin (2014) dan Iswari, et al. (2016). Oliviera, et al (2006) menggunakan *Simulated Annealing* dengan algoritma *Hill Climbing Strategy with Random Restart (Multi-Start)* untuk penyelesaian VRPTW. Sedangkan, Yu dan Lin (2014) menggunakan algoritma *restart simulated annealing* untuk kasus *location routing problem with simultaneous pickup and delivery*. Kemudian Iswari, et al (2016) menggunakan algoritma *restart simulated annealing* untuk penyelesaian permasalahan *blood pick-up routing*. Algoritma *restart simulated annealing* dari Iswari, et al. (2016) menjadi algoritma dasar untuk penelitian ini. Perbedaan penelitian ini dengan penelitian-penelitian sebelumnya yang menggunakan *simulated annealing* dengan strategi restart adalah adanya algoritma *hybrid* atau penggabungan dari algoritma *restart simulated annealing* dengan algoritma *Vehicle Neighborhood Search*. Sebelumnya pernah dilakukan *hybrid* antara algoritma *Vehicle Neighborhood Search* dengan algoritma *Tabu Search* untuk penyelesaian kasus CVRP oleh Halim dan Yu (2016). Pada penelitian ini, algoritma *Vehicle Neighborhood Search* digunakan pada tahap mencari solusi baru (*neighborhood solution*) pada setiap iterasi di algoritma *simulated annealing*.

Hybrid Restart Simulated Annealing with Variable Neighborhood Search (HRSA-VNS)

Algoritma *simulated annealing* (SA) pertama kali diperkenalkan oleh Metropolis, et al (1953). SA merupakan salah satu jenis algoritma *local search* yang bermula dari sebuah solusi awal (*initial solution*). Pada penelitian ini solusi awal dibangun menggunakan algoritma *Nearest Neighborhood*. Pada algoritma *Nearest Neighborhood*, kendaraan akan memulai dari sebuah dummy *depot* dan secara berulang mengunjungi pelanggan terdekat sampai kapasitas kendaraan tersebut habis. Algoritma kemudian membangun kembali rute yang baru dengan metode yang sama hingga semua pelanggan dapat dikunjungi.

Solusi pada algoritma ini direpresentasikan dengan untaian atau rangkaian angka yang terdiri dari permutasi dari sejumlah *n* pelanggan yang dinotasikan dengan {*1,2,...n*} dan N_{dummy} nol. Disini N_{dummy} adalah variabel yang menunjukkan dummy *depot* yang merupakan pertanda dalam pembentukan rute baru karena batasan-batasan yang ada.

Pada penelitian ini terdapat sejumlah mekanisme untuk *neighborhood moves*, yaitu

Swap, *Insert* dan *Reverse*. Untuk mekanisme *Swap*, dilakukan dengan cara memilih 2 *nodes* secara random kemudian menukar posisi dari kedua *nodes* tersebut. Pada mekanisme *Insert,* dipilih satu *node* secara random kemudian menyisipkan *node* tersebut diantara *nodes* yang lain. Kemudian untuk mekanisme *Reverse,* dipilih 2 *nodes* secara random lalu melakukan menukar urutan antara kedua *nodes* yang dipilih. Contoh dari ketiga mekanisme tersebut dapat dilihat pada Gambar 1-3. Untuk pemilihan mekanisme yang akan digunakan dari ketiga mekanisme tersebut, penelitian ini mengadopsi cara dari metode *Variable Neighborhood Search*. Dalam *variable neighborhood search*, akan dilakukan pencarian *neighborhood* dengan mengubah *neighborhood set* secara sistematis. Ide dibalik adanya penggunaan mekanisme pengubahan *neighborhood set* secara sistematis ini adalah untuk membantu algoritma dalam mencari porsi yang lebih luas dalam ruang pencarian. Tabel 1 menunjukkan *neighborhood set* yang dipakai dalam penelitian ini. Setiap iterasinya akan dibangun solusi baru berdasarkan P_{ij} yang merupakan probabilitas untuk memilih jenis *neighborhood moves* dari set i dengan probabilitas ke-j.

A	0	5	4	9	0	3	8	7	6	10	2	1	0
B	0	5	6	9	0	3	8	7	4	10	2	1	0

Gambar 1. *Swap*

A	0	5	4	9	0	3	8	7	6	10	2	1	0
B	0	5	9	0	3	8	7	6	4	10	2	1	0

Gambar 2. *Insert*

A	0	5	4	9	0	3	8	7	6	10	2	1	0
B	0	5	6	7	8	3	0	9	4	10	2	1	0

Gambar 3. *Reverse*

Keterangan :
A = Sebelum *neighborhood move*
B = Setelah *neighborhood move*

Tabel 1. *Neighborhood Set*

P_{ij}		*j*		
		0.3	**0.7**	**1**
		1	**2**	**3**
i	**1**	*Reverse*	*Insert*	*Swap*
	2	*Swap*	*Reverse*	*Insert*
	3	*Insert*	*Swap*	*Reverse*
	4	*Insert*	*Reverse*	*Swap*
	5	*Reverse*	*Swap*	*Insert*
	6	*Swap*	*Insert*	*Reverse*

Dapat dilihat pada Tabel 1 bahwa terdapat 6 jenis *neighborhood set* yang akan digunakan. Contohnya pada iterasi ke 1 dibangun bilang random 0.6, maka kita akan menggunakan *neighborhood set* yang pertama *i=1* dengan *j=2* karena angka 0.6 berada pada diantara 0.3 – 0.7, sehingga kita akan menggunakan *Insert moves*. Ketika sampai iterasi ke 7 maka akan kembali ke set pertama yaitu *i=1* dan seterusnya hingga iterasi terakhir.

Penelitian menggunakan startegi restart untuk mencegah solusi terjebak pada titik lokal optimal. Stategi restart memungkinkan algoritma untuk memulai lagi dari temperatur awal ketika terjadi sejumlah solusi yang tidak meningkat saat terjadi penurunan suhu. Gambar 4 berikut menunjukkan algoritma HRSA-VNS yang digunakan.

Algoritma HRSA-VNS
Mencari solusi terbaik (*X*)

Inputs:
Set dari operator *neighborhood moves*
Temperatur awal (T_0)
Cooling rate (α)
Jumlah iterasi (N_{iter})
$N_{non\text{-improving}}$
Jumlah restart maksimum (R_{max})
Initialization:
Bangkitkan solusi awal X_0;
While *stopping criteria* belum terpenuhi (belum mencapai jumlah R_{max})
repeat
Masukkan nilai temperatur awal (T_0)
repeat
Y←Bangkitkan solusi baru dari *neighborhood N(X)* menggunakan VNS
Hitung **Δ=** *F(Y)-F(X)*
if Δ < 0 ***then***
X← *Y*
if *F(X)<Fbest* ***then***
Fbest←*f(X)*
else
Bangkitkan bilangan random = *rand*(0,1)
if *rand* < exp(-Δ/*T)* ***then***
X← *Y*
Until jumlah iterasi (I_{iter}) tercapai
Lakukan *Local search* (*X*)
Lakukan penurunan temperatur sesuai *cooling procedure*
Until Jumlah *Fbest* yang tidak semakin baik mencapai $N_{non\text{-improving}}$
Return *X*

Gambar 4. Algoritma HRSA-VNS

Pertama, suhu awal ditentukan sebagai T_0 dan solusi awal *X* dihasilkan. Nilai fungsi tujuan dari *X* dinotasikan sebagai *obj (X).* Solusi terbaik *Xbest* saat ini adalah *X* karena baru masuk ke iterasi pertama. *Fbest* menunjukkan nilai fungsi tujuan terbaik yang diperoleh dan diinisialisasi sebagai *obj (X).*

Pada setiap iterasi, solusi *Y* di *N(X)* dipilih dan nilai fungsi tujuannya dievaluasi. Solusi Y akan dipilih dengan menggunakan skema

algoritma VNS. Jika *Y* lebih baik dari *X*, maka *Y* akan menggantikan *X* sebagai solusi baru. Jika tidak, *Y* diterima sebagai solusi baru dengan suatu probabilitas. Probabilitas ini dihitung dengan menggunakan fungsi Boltzmann, yaitu *p = exp (-E/T)* dengan *E = obj (Y) - obj (X).*

Setelah iterasi mencapai jumlah N_{iter}, suhu akan menurun. Penurunan suhu ini disebut mekanisme pendinginan (*cooling procedure*). *Cooling procedure* yang dipakai adalah $T=T_0*\alpha$ Setiap kali dilakukan penurunan suhu maka akan dilakukan *local search* dengan menggunakan algoritma 2opt. $N_{non-improving}$ adalah jumlah maksimum terjadinya pengurangan suhu di mana nilai fungsi tujuan terbaik tidak semakin meningkat. Kemudian, algoritma ini akan memulai kembali dari awal dengan suhu awal T_0 jika jumlah $N_{non-improving}$ telah tercapai. Algoritma akan berakhir ketika algoritma mencapai jumlah restart maksimum (R_{max}). Setelah algoritma selesai maka solusi terbaik akan diperoleh dari *Xbest*.

Hasil dan Pembahasan

Sebelum melakukan running model HRSA-VNS, perlu dilakukan penentuan parameter untuk menentukan nilai parameter yang terbaik (Iswari et al., 2016). Nilai parameter memiliki kemungkinan untuk mempengaruhi kualitas hasil komputasi. Oleh karena itu, sebelum menjalankan algoritma, pengaturan parameter dilakukan untuk menemukan parameter terbaik yang dapat menghasilkan solusi yang paling optimal. Setting parameter dalam penelitian ini dilakukan dengan menggunakan 2^k desain faktorial.

2^k desain faktorial berhasil digunakan sebagai prosedur untuk menemukan pengaturan parameter yang efektif di heuristik (Coy et al., 2001). Ada lima parameter dalam algoritma HRSA-VNS yang dipilih oleh desain faktorial 2k, yaitu, N_{iter}, α, T_0, $N_{non-improving}$, dan R_{max}. N_{iter} menunjukkan jumlah iterasi hasil pencarian pada suhu tertentu. α adalah koefisien yang mengendalikan mekanisme pendinginan. Sementara T_0 merupakan suhu awal, $N_{non-improving}$ adalah jumlah maksimum pengurangan suhu di mana *best solution* tidak meningkat. Algoritma akan melakukan restart dengan suhu awal jika $N_{non-improving}$ tercapai. R_{max} adalah jumlah maksimum algoritma bisa melakukan restart, parameter ini digunakan untuk menjadi stopping criteria dalam algoritma ini.

Sebelum melakukan pengaturan parameter, kita melakukan analisis sensitivitas menggunakan metode *One At a Time* (OAT) untuk mengidentifikasi level atas dan bawah untuk 2^k desain faktorial. Langkah pertama dimulai dengan menentukan nilai untuk setiap parameter yang akan diuji. Nilai awal dari parameter yang diuji dipilih secara acak. Analisis sensitivitas dilakukan untuk semua parameter dengan memvariasikan satu parameter ketika parameter lainnya sama. Hasil OAT kemudian digunakan sebagai sebagai level bawah dan batas pada 2^k desain faktorial. Setiap parameter memiliki 4 level nilai yang didapat dari penelitian sebelumnya (Iswari et al., 2016). Berikut ini adalah level nilai dari kelima parameter yang digunakan.

a. N_{iter}= 3000, 5000, 7000, 9000
b. α = 0,85, 0,90, 0,95, 0,99
c. T_0 = 10, 20, 50, 80
d. $N_{non-improving}$ = 100, 150, 200, 250
e. R_{max} = 3, 5, 7, 10

Nilai awal dalam penelitian ini diambil secara acak, yaitu N_{iter}= 3000, α = 0,95, T_0 = 10, $N_{non-improving}$ = 100, dan R_{max} = 10. Hasil rata-rata dari pengujian awal dapat dilihat pada Tabel 2.

Tabel 2. Hasil OAT

Para-meter	N_{iter}	α	T_0	$N_{non-improving}$	R_{max}	Rata-rata
N_{iter}	3000	0.95	10	100	10	1.460
	5000	0.95	10	100	10	1.446
	7000	0.95	10	100	10	**1.405**
	9000	0.95	10	100	10	**1.387**
α	3000	**0.85**	10	100	10	**1.437**
	3000	0.9	10	100	10	1.466
	3000	**0.95**	10	100	10	**1.460**
	3000	0.99	10	100	10	1.468
T_0	3000	0.95	10	100	10	1.460
	3000	0.95	20	100	10	1.477
	3000	0.95	**50**	100	10	**1.425**
	3000	0.95	**80**	100	10	**1.443**
$N_{non-improving}$	3000	0.95	10	100	10	1.460
	3000	0.95	10	**150**	10	**1.455**
	3000	0.95	10	200	10	1.465
	3000	0.95	10	**250**	10	**1.383**
R_{max}	3000	0.95	10	100	**3**	**1.429**
	3000	0.95	10	100	5	1.440
	3000	0.95	10	100	**7**	**1.386**
	3000	0.95	10	100	10	1.460

Dari hasil OAT didapatkan 2 nilai terbaik dari setiap parameternya yang kemudian menjadi nilai level *low* dan *high* dari 2^k *factorial design*. 2 level yang digunakan dapat dilihat pada Tabel 3.

Tabel 3. Hasil Level Terpilih

Parameter	Low	High
N_{iter}	7000	9000
α	0.85	0.95
T_0	50	80
$N_{non-improving}$	150	250
R_{max}	3	7

Selanjutnya, eksperimen desain 2^k dilakukan untuk mengidentifikasi kombinasi parameter yang terbaik. Setiap kombinasi

parameter dievaluasi untuk dengan melihat hasil rata-rata nilai fungsi tujuan yang dihasilkan pada Tabel 4.

Tabel 4. Hasil 2^k desain faktorial

No	N_{iter}	α	T_0	$N_{non-improving}$	R_{max}	Rata-rata
1	7000	0.85	50	150	3	1396,743
2	7000	0.85	50	150	7	1399,922
3	7000	0.85	50	250	3	1387,724
4	7000	0.85	50	250	7	1372,696
5	7000	0.85	80	150	3	1412,870
6	7000	0.85	80	150	7	1412,582
7	7000	0.85	80	250	3	1372,476
8	7000	0.85	80	250	7	1389,001
9	7000	0.95	50	150	3	1430,874
10	7000	0.95	50	150	7	1407,460
11	7000	0.95	50	250	3	1447,714
12	7000	0.95	50	250	7	1389,350
13	7000	0.95	80	150	3	1390,650
14	7000	0.95	80	150	7	1402,218
15	7000	0.95	80	250	3	1383,761
16	7000	0.95	80	250	7	1381,867
17	9000	0.85	50	150	3	1391,234
18	9000	0.85	50	150	7	1463,694
19	9000	0.85	50	250	3	1404,330
20	9000	0.85	50	250	7	1445,411
21	9000	0.85	80	150	3	1364,418
22	9000	0.85	80	150	7	1398,625
23	9000	0.85	80	250	3	1410,099
24	9000	0.85	80	250	7	1365,603
25	9000	0.95	50	150	3	1390,813
26	9000	0.95	50	150	7	1386,662
27	9000	0.95	50	250	3	1400,115
28	**9000**	**0.95**	**50**	**250**	**7**	**1345,452**
29	9000	0.95	80	150	3	1377,055
30	9000	0.95	80	150	7	1396,932
31	9000	0.95	80	250	3	1369,344
32	9000	0.95	80	250	7	1406,987

Hasil terbaik didapatkan dari kombinasi yang menghasilkan nilai rata-rata total jarak yang paling kecil. Di sini didapatkan pada kombinasi ke 28 dengan nilai N_{iter}= 9000, α = 0,95, T0 = 50, $N_{non-improving}$ = 250, dan R_{max}= 7.

Setelah ditentukan parameter terbaik untuk algoritma HRSA-VNS maka dilakukan eksperimen untuk mengetes performa dari algoritma yang telah dibangun dengan menggunakan data *benchmark* VRPTW dari Solomon (1988). Dataset *benchmark* dari Solomon memiliki jumlah titik sebesar 100 dan terdiri dari enam jenis masalah, yaitu C1, C2, R1, R2, RC1, dan RC2. Huruf C, R, RC mewakili tiga jenis pelanggan. Huruf C menunjukkan pelanggan yang memiliki data yang berkelompok (*cluster*). Kemudian huruf R menunjukkan pelanggan dengan data geografis yang random. Sedangkan huruf RC berisi data campuran dari data berkelompok (*cluster*) dan random. Kemudian selain huruf terdapat juga angka 1 dan 2 yang menunjukkan jenis *time horizon*-nya. Angka 1 menunjukkan waktu penjadwalan yang pendek atau dengan kata lain kendaraan hanya dapat mengunjungi sedikit *customer* dalam satu rute. Sedangkan angka 2 menunjukkan waktu horizon penjadwalan yang panjang dengan sehingga dapat mengunjungi banyak *customer* dalam satu rute.

Hasil dari performa algoritma HRSA-VNS dapat dilihat pada Tabel 5. Pada Tabel 5, algoritma yang dibangun dibandingkan dengan BKS *(Best Known Solution)* dan algoritma SA biasa. TD *(total distance)* merupakan total jarak yang dihasilkan sedangkan NV *(number of vehicle)* adalah jumlah kendaraan yang dibutuhkan, sedangkan *diff* atau *difference* adalah persentase hasil pengurangan total jarak HRSA-VNS dan BKS per nilai BKS. Nilai *diff* yang negatif menunjukkan bahwa HRSA-VNS menghasilkan hasil yang lebih baik dan sebaliknya jika nilai *diff* postif maka hasil BKS yang lebih baik.

Tabel 5. Hasil Performansi HRSA-VNS vs BKS

Dataset	BKS[a]		HRSA-VNS		
	TD	NV	TD	NV	diff[a]
C101	828,9[c]	10	828,9	10	0,00%
C201	591,6[c]	3	591,6	3	0,00%
R101	1645,8[d]	19	1658,1	19	0,75%
R201	1252,4[e]	4	1199,9	9	-4,18%
RC101	1696,9[f]	14	1670,2	16	-1,58%
RC201	1406,9[g]	4	1309,0	10	-6,96%
Rata-rata	**1237,1**	**9,0**	**1209,6**	**11,2**	**-2,00%**
Dataset	BKS[b]		HRSA-VNS		
	TD	NV	TD	NV	diff[b]
C101	828,9	10	828,9	10	0,00%
C201	591,6	3	591,6	3	0,00%
R101	1642,9	19	1658,1	19	0,93%
R201	1148,5	9	1199,9	9	4,48%
RC101	1623,6	15	1670,2	16	2,87%
RC201	1274,5	9	1309,0	10	2,70%
Rata-rata	**1184,9**	**10,8**	**1209,6**	**11,2**	**1,83%**

Keterangan :
a = BKS dari web Solomon (2005)
b = BKS dari Alvarenga, et al. (2007) - *A genetic and set partitioning two-phase*
c = Rochat dan Taillard (1995) - *Probabilistic dan diverfisied local search*
d = Homberger (2000) - *Distributed-parallel Metaheuristics*
e = Homberger dan Gehring (1999) - *Two Evolutionary Metaheuristics*
f = Taillard, et al. (1997) - *Tabu Search*
g = Mester (2002) - *An Evolutionary Strategies Algorithm*

Dari Tabel 5, dapat dilihat bahwa algoritma HRSA-VNS mempunyai performa yang baik dan cukup kompetitif dengan metode-metode yang lain. Hal ini dapat dilihat

dari hasil rata-rata *difference*-nya. BKS dari website Solomon mengambil hasil terbaik dari beberapa peneliti dengan metode penyelesaian yang berbeda-beda. Apabila dibandingkan dengan *Best Known Solution* dari Web Solomon didapatkan hasil total jaraknya lebih kecil, akan tetapi jumlah kendaraan yang digunakan lebih banyak. BKS dari website Solomon memiliki 2 fungsi tujuan yaitu meminimasi total kendaraan dan meminimasi jarak tempuh padahal pada penelitian ini fungsi objektifnya hanyalah untuk meminimasi jarak tempuh. Oleh karena itu dilakukan juga perbandingan dengan BKS dari Alvarenga yang menggunakan metode *hybrid genetic algorithm* dengan *column generation*. Dari hasil yang didapatkan didapatkan selisih atau *difference* yang relatif kecil yaitu 1,83% atau 0,0183.

Selain membandingkan dengan BKS, algoritma HRSA-VNS juga dibandingkan dengan metode SA biasa tanpa ada modifikasi apapun. Hasil perbandingannya dapat dilihat pada Tabel 6.

Tabel 6. Hasil Performansi HRSA-VNS vs SA

Dataset	SA		HRSA-VNS		*Diff*
	TD	NV	TD	NV	
C101	828,9	10	828,9	10	0,00%
C201	650,6	4	591,6	3	-9,07%
R101	1.671,7	19	1658,1	19	-0,81%
R201	1.204,6	9	1199,9	9	-0,38%
RC101	1.690,4	16	1670,2	16	-1,20%
RC201	1.332,2	10	1309,0	10	-1,74%
Rata-rata	1229,7	11,3	1209,6	11,2	-2,20%

Dari Tabel 5, didapatkan hasil bahwa algoritma HRSA-VNS yang dibangun ternyata menghasilkan hasil yang lebih baik dibandingkan SA biasa tanpa modifikasi. Hal ini dapat dilihat dari rata-rata *difference* yang lebih kecil karena HRSA-VNS dapat menghasilkan total jarak tempuh yang lebih kecil. Adanya peningkatan hasil ini didapatkan karena terdapat mekanisme-mekanisme yaitu strategi restart, *local search*, dan pencarian solusi terdekat dengan algoritma VNS. Mekanisme ini dilakukan untuk membuat diversifikasi pencarian dengan menghindari terjebak dalam solusi *local optimal*. Algoritma HRSA-VNS yang dibangun diharapkan dapat menjelajahi porsi yang lebih besar dari ruang pencarian dengan harapan dapat menemukan solusi global optimal.

Kesimpulan dan Saran

Penelitian ini mengambil objek permasalahan penentuan rute optimal dengan adanya *constraint time windows* atau yang lebih dikenal sebagai *Vehicle Routing Problem with Time Windows* (VRPTW). Fungsi objektif dari permasalahan VRPTW adalah untuk meminimalkan total jarak yang ditempuh kendaraan. Dalam penyelesaian kasus VRPTW, penelitian ini menggunakan suatu metode metode metaheuristik yaitu *Hybrid Restart Simulated Annealing with Variable Neighborhood Search* (HRSA-VNS). Algoritma HRSA-VNS memodifikasi algoritma *Simulated annealing* (SA) dengan menambahkan strategi restart dan menggunakan skema algoritma VNS dalam tahap pencarian solusi tetangga (*neighborhood search*). Pada penelitian ini dilakukan eksperimen untuk menguji performa dari algoritma yang telah dibangun dengan menggunakan data VRPTW benchmark dari Solomon (1987). Pengujian performa algoritma yang dibangun dilakukan dengan membandingkan hasil HRSA-VNS dengan BKS *(Best Known Solution)* dan algoritma SA biasa tanpa modifikasi. Dari hasil yang didapatkan, diketahui bahwa algoritma yang dibangun cukup baik dalam menyelesaikan kasus VRPTW dan dapat bersaing dengan algoritma-algoritma yang lain. Namun, algoritma yang dibangun saat ini masih dapat dikembangkan sehingga dapat menghasilkan jarak tempuh maupun jumlah kendaraan yang lebih kecil dibanding algoritma-algoritma lain yang sudah ada.

Daftar Pustaka

Alfa, A. S., S. S. Heragu dan M. Chen (1991). A 3-opt based simulated annealing algorithm for vehicle routing problems. *Computers & Industrial Engineering* 21(1): 635-639.

Alvarenga, G. B., G. R. Mateus dan G. de Tomi (2007). A genetic and set partitioning two-phase approach for the Vehicle Routing Problem with Time Windows. *Computers & Operations Research*, 34(6), 1561-1584.

Breedam, A. V. (1995). Improvement heuristics for the Vehicle Routing Problem based on Simulated annealing. *European Journal of Operational Research*, 86, 480-490.

Dantzig, G. B. dan J. H. Ramser (1959). The Truck Dispatching Problem.*Management Science*, 6(1): 80-91.

Halim, C. dan Yu, V. F. (2016). Minimum Cost Vertex-Disjoint Path Cover Problem. *Electronic Thesis and Dissertation (ETD) National Taiwan University of Science and Technology.* Diakses dari : http://pc01.lib.ntust.edu.tw/ETD-db/ETD-search/view etd?URN=etd-0115116-143744 [2017, 1 Maret]

Homberger, J. (2000). *Verteilt-parallele Metaheuristiken zur Tourenplanung*, Gaber, Wiesbaden

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

Iswari, T., Yu, V.F., dan Asih, A.M.S. (2016). A Simulated Anealing Heuristic for Blood Pick-Up Routing Problem. *Electronic Thesis and Dissertation (ETD) National Taiwan University of Science and Technology.* Diakses dari: https://pc01.lib.ntust.edu.tw/ETD-db/ETD-search/view_etd?URN=etd-0620116-133237 [2017, 1 Maret].

Kumar,S.N., dan Paneerselvam, R., (2012), A Survey on The Vehicle Routing Problem and Its Variants, *Intelligent Information Management*, 4, 66-74.

Laporte, G. (1992). The Vehicle Routing Problem: An overview of exact and approximate algorithms. *European Journal of Operational Research* 59(3), 345-358.

Lin, S.-W., K.-C. Ying, Z.-J. Lee dan F.-H. Hsi (2006). Applying Simulated annealing Approach for Capacitated Vehicle Routing Problems. *2006 IEEE International Conference on Systems, Man, and Cybernetics*, 639-644.

Lin, S.-W., V. F. Yu dan S. Y. Chou (2011). A simulated annealing heuristic for the truck and trailer routing problem with time windows. *Expert Systems with Applications*, 38(12), 15244-15252.

Mester, D. (2002). An Evolutionary Strategies Algorithm for Large Scale Vehicle Routing Problem with Capacitate and Time Windows Restrictions, *Working Paper Institute of Evolution*, *University of Haifa, Israel*.

Metropolis, N., A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller dan E. Teller (1953). Equation of State Calculations by Fast Computing Machines. *The Journal of Chemical Physics,* 21(6), 1087.

Oliviera, H. C. B., G. C. Vasconcelos dan G. B. Alvarenga. (2006). A Multi-Start Simulated Annealing Algorithm for the Vehicle Routing Problem with Time Windows. *Proceedings of the Ninth Brazilian Symposium on Neural Networks*.

Rochat, Y. dan Taillard, E.D. (1995). Probabilistic Diversification and Intensification in Local Search for Vehicle Routing. *Journal of Heuristics*, 1, 147-167.

Solomon, M. M. dan J. Desrosiers. (1988). Survey Paper—Time Window Constrained Routing and Scheduling Problems.*Transportation Science*, 22(1), 1-13.

Solomon. M. M (2005). *Best Known Solution Identified by Heuristic*. Diunduh dari : http://web.cba.neu.edu/~msolomon/heuristi.htm

Taillard,E., Badeau, P., Gendreau, M., Geurtin, F. dan Potvin, J.Y. (1997). A Tabu Search Heuristic for the Vehicle Routing Problem with Time Windows. *Transportation Science*, 31, 170-186

Xiao, Y., Zhao, Q., Kaku I., dan Xu, Y. (2012). Development of a fuel consumption optimization model for the capacitated vehicle routing problem. *Computers & Operations Research*,39(7), 1419-1431.

Xie F., Ji, S., Liu Y., dan Huang, X., 2008, Research of Coupling Relation between City Logistics and Economy based on Artificial Neural Network, *Institute of Electrical and Electronics Engineers (IEEE) Journal.*

Yu, V. F. dan S.-W. Lin (2014). Multi-start simulated annealing heuristic for the location routing problem with simultaneous pickup and delivery. *Applied Soft Computing* 24: 284-290.

Yu, V. F., S.-W. Lin, W. Lee dan C.-J. Ting (2010). A simulated annealing heuristic for the capacitated location routing problem.*Computers & Industrial Engineering*, 58(2) 288-299.