JKMA
Jurnal Kajian Matematika dan Aplikasinya
VOLUME 2 NOMOR 1, Januari 2021
e-ISSN: 2722-7650. http://journal2.um.ac.id/index.php/jkma

ALGORITMA VARIABLE NEIGHBORHOOD DESCENT (VND) PADA VEHICLE ROUTING PROBLEM WITH SIMULTANEOUS DELIVERY AND PICKUP (VRPSDP) DAN IMPLEMENTASINYA

Yulio Christopher[1], Sapti Wahyuningsih[1, *], Darmawan Satyananda[1]
[1]Jurusan Matematika, FMIPA, Universitas Negeri Malang

Email : yuliochristo@gmail.com (*Y. Christopher*), sapti.wahyuningsih.fmipa@um.ac.id (*S. Wahyuningsih*), darmawan.satyananda.fmipa@um.ac.id (*D. Satyananda*)
* Corresponding Author

Abstract
Vehicle Routing Problem with Simultaneous Delivery and Pickup (VRPSDP) is a variant of Vehicle Routing Problem (VRP). VRPSDP has special constraints, namely requests and returns are carried out simultaneously. In this article we will use the Variable Neighborhood Descent (VND) algorithm to solve VRPSDP problems. There are two steps taken to use the VND algorithm on VRPSDP, namely finding an initial solution with the Insertion Heuristic algorithm and improving the position of the customer by using the neighborhood operator on the VND algorithm. The implementation of the VND algorithm on VRPSDP has been successfully made using the Borland Delphi 7.0 programming language. Inputs contained in the program are point position, distance between points, customer requests and returns and vehicle capacity. The output contained in the program in the form of routes that have been completed using an algorithm and output in the form of images of the final solution that has been obtained. Based on the results obtained, an example with 6 customers produces 3 routes with a total distance of 266 km, while an example with 10 customers produces 4 routes with a total distance of 100 km.

Keywords*: distribution, graph, Vehicle Routing Problems with Simultaneous Delivery and Pickup (VRPSDP), Variable Neighborhood Descent (VND)*

Submited: 04 October 2020; Revised: 10 November 2020; Accepted Publication: 14 December 2020; Published Online: January 2021
DOI: 10.17977/um055v2i1p5-13

PENDAHULUAN

Dalam kehidupan sehari-hari, sering ditemukan permasalahan mengenai pendistribusian barang. Distribusi adalah suatu kegiatan memasarkan produk dari produsen ke konsumen. Kegiatan distribusi merupakan salah satu faktor penting dalam suatu perusahaan untuk memperoleh keuntungan. Salah satu cara untuk memperoleh keuntungan tersebut adalah dengan meminimalkan biaya distribusi. Biaya distribusi dapat diminimalkan dengan memilih rute distribusi yang tepat. Permasalahan distribusi dapat diselesaikan menggunakan salah satu disiplin ilmu yaitu matematika. Salah satu cabang ilmu matematika yang dapat menyelesaikan masalah distribusi adalah teori *graph*, khususnya pada materi *Vehicle Routing Problem* (VRP).

Menurut (Simsir and Ekmekci, 2019), VRP merupakan permasalahan untuk mencari rute terbaik suatu kendaraan yang akan keluar dari satu atau lebih depot untuk melakukan pengiriman dan pengembalian barang, di mana lokasi *customer* tersebar secara luas. Solusi yang diharapkan dari penyelesaian VRP adalah dapat memberikan keuntungan yang signifikan dengan mengurangi biaya pengeluaran. Pada perkembangannya, VRP memiliki banyak varian dengan berbagai kendala, salah satunya *Vehicle Routing Problem with Simultaneous Delivery and Pickup* (VRPSDP). VRPSDP merupakan varian VRP yang memiliki kendala khusus yaitu permintaan dan pengembalian dilakukan secara bersamaan. Hal ini menyebabkan fluktuasi muatan kendaraan saat ini yang mengakibatkan peningkatan kesulitan dalam memeriksa kelayakan solusi. Oleh karena itu, aspek utama adalah memeriksa muatan kendaraan saat ini pada setiap *customer* karena kapasitas kendaraan tidak boleh berlebih (Zhu et al., 2017).

Permasalahan VRPSDP merupakan permasalahan yang sering terjadi dalam kegiatan logistik pada kehidupan nyata. Kendala khusus pada varian VRPSDP adalah bahwa semua barang yang akan dikirim ke *customer*, dikirim dari depot dan semua barang yang akan diterima dari *customer* akan dikirim kembali ke depot yang sama. Permintaan dan pengembalian barang dilakukan secara bersamaan (Simsir and Ekmekci,

2019). Tujuan dari VRPSDP adalah meminimalkan total jarak dari rute yang dilalui kendaraan ketika mendistribusikan barang tanpa melanggar kendala kapasitas.

Permasalahan VRPSDP telah banyak diselesaikan dengan menggunakan berbagai macam algoritma, baik algoritma *heuristic* maupun algoritma *metaheuristic*. Beberapa penelitian yang terkait dengan VRPSDP dan algoritma yang digunakan antara lain algoritma *Artificial Bee Colony* (ABC) yang dibahas oleh (Simsir and Ekmekci, 2019), algoritma *Multiple Neighborhood Guided Local Search* (MN_GLS) yang dibahas oleh (Zhu et al., 2017), algoritma *Ant Colony System-Variable Neighborhood Search* (ACS-VNS) yang dibahas oleh (Kalayci and Kaya, 2016), algoritma *Parallel Simulated Annealing* yang dibahas oleh (Mu et al., 2016), dan algoritma *Effective Ant Colony Optimization* (EACO) yang dibahas oleh (Sayyah et al., 2016).

Algoritma *Variable Neighbourhood Descent* (VND) merupakan algoritma metaheuristik yang diterapkan pada beberapa masalah optimasi kombinatorial (Harzi and Krichen, 2017). Algoritma VND didasarkan pada gagasan bahwa eksplorasi dari solusi yang diberikan harus dimulai dengan *neighborhood* yang telah ditentukan. *Neighborhood* merupakan himpunan hasil solusi dari penerapan suatu operator yang telah ditentukan. Jika tidak ada perbaikan yang ditemukan, maka pencarian harus diperluas ke *neighborhood* yang berbeda. Hal ini berlanjut sampai pencarian mencapai *neighborhood* terbesar yang tersedia. Jika pada suatu titik terdapat solusi yang lebih baik, maka prosesnya diatur ulang ke *neighborhood* pertama. Di sisi lain, proses berhenti jika semua *neighborhood* telah dieksplorasi dan tidak ada perbaikan ditemukan (Herrán et al., 2019). Beberapa penelitian yang menggunakan algoritma VND untuk menyelesaikan suatu permasalahan di antaranya *Vehicle Routing Problem with Time Windows* (Harzi and Krichen, 2017), *Capacitated Vehicle Routing Problem* (Amous et al., 2017), dan pada CVRP, MDVRP, dan VRPTW (Satyananda and Wahyuningsih, 2019).

Berdasarkan uraian di atas, maka pada artikel ini akan dikaji mengenai penggunaan algoritma VND untuk menyelesaikan VRPSDP dan implementasi program dengan bahasa pemrograman *Borland Delphi 7.0*. Selanjutnya, program yang telah dibuat akan digunakan untuk menyelesaikan contoh VRPSDP dengan menggunakan 6 *customer* dan 10 *customer*.

METODE

Metode yang digunakan pada penelitian ini adalah sebagai berikut:

1. Melakukan studi pustaka untuk mengkaji penerapan algoritma VND pada permasalahan VRPSDP.
2. Mengimplementasikan algoritma VND pada VRPSDP ke program komputer menggunakan *Borland Delphi 7*, dengan langkah menginputkan data, memproses penyelesaian dengan algoritma, dan selanjutnya akan diperoleh solusi akhir algoritma.
3. Menerapkan contoh permasalahan VRPSDP dengan menggunakan 6 *customer* dan 10 *customer* pada program yang telah dibuat.

HASIL DAN PEMBAHASAN

Algoritma VND pada VRPSDP

Dalam penyelesaiannya, algoritma VND pada VRPSDP memiliki dua langkah yang digunakan yaitu mencari solusi awal dan memperbaiki posisi *customer* dengan menggunakan operator *neighborhood* pada algoritma VND. Pada tahap pencarian solusi awal, akan digunakan algoritma *Insertion Heuristic*, lalu akan dilakukan perbaikan posisi *customer* untuk mencari total jarak minimum tanpa melanggar kendala kapasitas. Dalam melakukan perbaikan posisi *customer*, akan digunakan enam operator *neighborhood*. Dari keenam operator *neighborhood* tersebut akan diperoleh beberapa solusi. Solusi terbaik akan dipilih dan akan dibandingkan dengan solusi awal. Jika semua operator telah dilakukan dan solusi perbaikan yang diperoleh konvergen ke suatu nilai minimum, maka kondisi optimum dari algoritma VND telah tercapai. Langkah-langkah algoritma VND pada VRPSDP adalah:

1. Mencari Solusi Awal

Pada tahap ini akan dicari solusi awal dari permasalahan VRPSDP. Solusi awal tersebut akan menjadi rute sementara. Pembangkitan solusi awal pada penelitian ini menggunakan algoritma *Insertion Heuristic*. Adapun langkah-langkah algoritma *Insertion Heuristic* (Lai et al., 2008) yaitu :

Fase I: Membentuk Rute dengan $d_i \geq p_i$

1. Membentuk rute awal $PR_k = (0,0)_k$ untuk setiap kendaraan f_k
2. Untuk setiap *customer* i dengan $d_i \geq p_i$, urutkan hasil dari $d_i - p_i$ dan buat daftar d-*customer* dari terbesar ke terkecil

3. Diawali dari daftar d-*customer* teratas, pilih *customer* i. Cari sisi (h, j) dan sisipkan i di antara (h, j) pada rute PR_k yang telah ada, sehingga akan diperoleh sisi baru yaitu (h, i) dan (i, j). Hitung nilai penyisipan jaraknya dengan cara $(c_{hi} + c_{ij} - c_{hj})$, di mana c merupakan jarak. Lalu, cek kendala kapasitas pada penyisipan jarak *customer* i. Pilih rute PR_k yang memiliki penyisipan jarak minimum dan tidak melanggar kendala kapasitas. Jika tidak melanggar kendala kapasitas dan memiliki nilai penyisipan jarak yang sama, maka dapat dipilih rute PR_k sebarang. Jika semua penyisipan jarak melanggar kendala kapasitas, maka sisipkan *customer* i pada rute PR_k yang baru.
4. Jika *customer* i sudah masuk ke dalam rute, hapus *customer* i dari daftar d-*customer*. Lalu ulangi langkah 3 sampai tidak ada *customer* pada daftar d-*customer*.

Fase II: Menyisipkan *Customer* dengan $d_i < p_i$

Untuk setiap *customer* i dengan $d_i < p_i$, sisipkan *customer* i ke semua rute PR_k yang telah terbentuk pada fase I. Hitung nilai penyisipan jarak dengan cara $(c_{hi} + c_{ij} - c_{hj})$. Cek kendala kapasitas di semua penyisipan. Pilih penyisipan jarak minimum dan tidak melanggar kendala kapasitas. Jika semua penyisipan jarak melanggar kendala kapasitas, maka sisipkan *customer* i pada rute PR_k yang baru. *Customer* i hanya boleh disisipkan satu kali. Proses berhenti ketika semua *customer* i telah masuk ke dalam rute.

2. Memperbaiki posisi *customer*

Pada tahap sebelumnya telah diperoleh solusi awal. Solusi awal tersebut belum dapat dipastikan sebagai hasil yang memiliki total jarak minimum. Maka dari itu, akan dilakukan perbaikan posisi *customer* dengan menggunakan beberapa operator *neighborhood* yang bertujuan untuk menemukan kemungkinan hasil yang lain yang lebih baik dari solusi awal. Beberapa operator *neighborhood* pada algoritma VND yang akan digunakan untuk memperbaiki posisi *customer* yaitu *swap 1-1, swap 2-1, swap 2-2, insertion, exchange,* dan *2-opt*. Total jarak yang didapatkan dari perbaikan posisi *customer* akan dibandingkan dengan total jarak yang didapatkan dari solusi awal.

Perbaikan posisi *customer* dilakukan dengan cara menukar *customer* antar rute (*inter-route*) dan menukar *customer* di dalam rute sendiri (*intra-route*). Berdasarkan (Harzi and Krichen, 2017), pertukaran posisi *customer* dilakukan dengan menukar secara *inter-route* terlebih dahulu yaitu dengan operator *swap 1-1, swap 2-1,* dan *swap 2-2* secara terurut. Lalu dilakukan pertukaran secara *intra-route* dengan operator *insertion*, *exchange*, dan *2-opt* secara terurut.

Proses perbaikan posisi *customer* dimulai dengan menyiapkan daftar *neighborhood* yang akan digunakan secara terurut yaitu DN_x = *swap 1-1, swap 2-1, swap 2-2, insertion, exchange, 2-opt* dan menghitung total jarak solusi awal $(f(b))$. Daftar *neighborhood* tersebut akan diberi notasi DN_1 sampai DN_6 secara berurutan. Perbaikan diawali dari DN_1. Perbaikan dengan DN_1 menggunakan solusi awal (b). Notasi untuk hasil perbaikan yaitu (b'). Lalu hitung total jarak setelah perbaikan $(f(b'))$. Jika $\big(f(b')\big) < (f(b))$ pada DN_1, maka (b') akan menjadi (b) yang baru dan akan diperbaiki lagi dengan DN_1. Jika $\big(f(b')\big) \geq (f(b))$, maka akan diperbaiki dengan DN_x berikutnya yaitu DN_2. Jika pada perbaikan DN_2 menghasilkan $\big(f(b')\big) < (f(b))$, maka (b') akan menjadi (b) yang baru dan akan diperbaiki lagi dengan DN_1. Jika pada perbaikan DN_2 menghasilkan $\big(f(b')\big) \geq (f(b))$, maka akan diperbaiki dengan DN_3. Pengerjaan dilakukan dengan cara yang sama hingga DN_6. Semua perbaikan tersebut harus memerhatikan kendala kapasitas. Proses perbaikan berhenti ketika telah melewati DN_6 dan telah mencapai kondisi optimum.

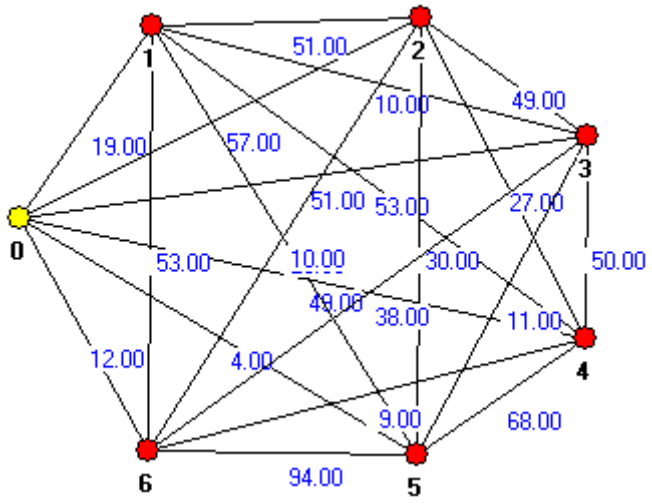
Secara sederhana, langkah algoritma VND pada VRPSDP pada artikel ini adalah sebagai berikut:

1. Mencari solusi awal (b) dengan algoritma *Insertion Heuristic*
2. Menyiapkan daftar *neighborhood* DN_x dan menghitung $f(b)$ dari solusi awal
3. Melakukan perbaikan dengan DN_x secara terurut, x dimulai dari 1
4. Proses di bawah akan diulang sampai $x = 6$
   a. Memperbaiki b dengan DN_x. Diperoleh hasil perbaikan b'
   b. Cari nilai $(f(b')$.
   c. Jika $\big(f(b')\big) < (f(b))$ maka $b' = b$ sehingga $f(b') = f(b)$ dan $x = 1$.
      Jika $\big(f(b')\big) \geq (f(b))$ maka $x = x + 1$.

Implementasi Algoritma VND pada VRPSDP

Implementasi algoritma VND pada VRPSDP telah dibuat dengan menggunakan bahasa pemrograman *Borland Delphi 7*. *Input* yang terdapat pada program adalah posisi titik, jarak antar titik, permintaan dan pengembalian *customer* serta kapasitas kendaraan. *Output* yang terdapat pada program berupa rute-rute yang

telah diselesaikan dengan menggunakan algoritma VND pada VRPSDP dan *output* yang berupa gambar dari solusi akhir yang telah diperoleh. Gambar 1 merupakan contoh VRPSDP dengan 6 *customer* dan 10 *customer* yang di selesaikan dengan menggunakan program yang telah dibuat.



(a)

Input Titik | Input Bobot | Parameter | Hasil | Graph Hasil

	0	1	2	3	4	5	6
0		19	57	51	49	4	12
1	19		51	10	53	25	53
2	57	51		49	27	30	10
3	51	10	49		50	11	38
4	49	53	27	50		68	9
5	4	25	30	11	68		94
6	12	53	10	38	9	94	

(b)

Input Titik | Input Bobot | Parameter | Hasil | Graph Hasil

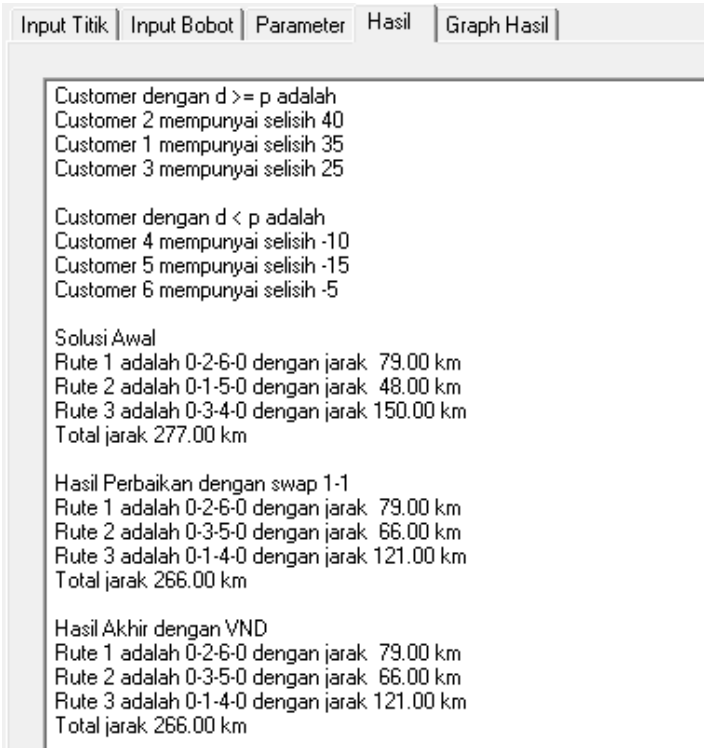
Customer	Permintaan
1	485
2	540
3	325
4	290
5	300
6	245

Customer	Pengembalian
1	450
2	500
3	300
4	300
5	315
6	250

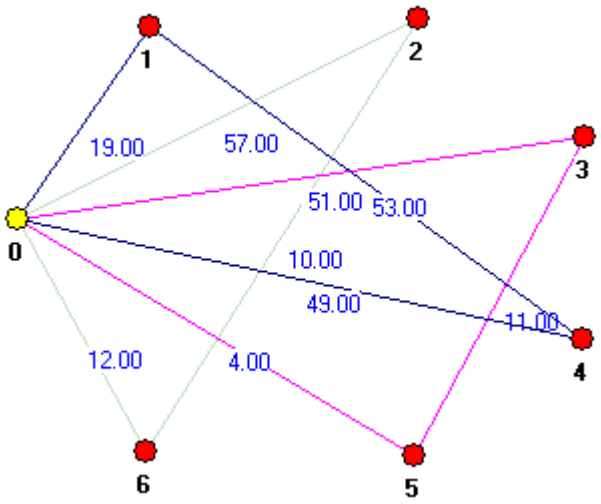
Kapasitas

800

(c)



(d)

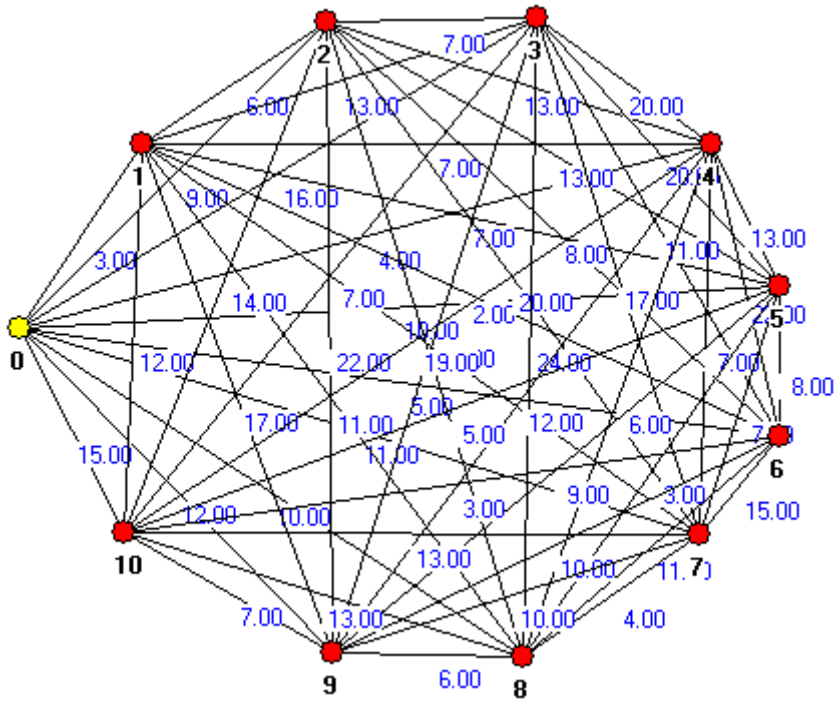


(e)

Gambar 1. Tampilan Penyelesaian VRPSDP dengan 6 *customer*. (a) *Graph* awal, (b) *Input* Bobot, (c) Input Permintaan, Pengembalian, dan Kapasitas, (d) Hasil akhir yang diperoleh, (e) Rute akhir

Berdasarkan hasil penyelesaian dengan program, hasil akhir diperoleh setelah mengalami perbaikan sebanyak 1 kali, di mana hasil akhirnya menghasilkan tiga rute. Rute pertama yaitu 0-2-6-0 dengan jarak 79 km, rute kedua yaitu 0-3-5-0 dengan jarak 66 km, dan rute ketiga yaitu 0-1-4-0 dengan jarak 121 km. Ketiga rute memiliki total jarak yaitu 266 km. Sedangkan tampilan program contoh VRPSDP dengan 10 *customer* dapat dilihat pada Gambar 2.

Berdasarkan hasil penyelesaian dengan program, hasil akhir diperoleh setelah mengalami perbaikan sebanyak 7 kali, di mana hasil akhirnya menghasilkan empat rute. Rute pertama yaitu 0-4-7-1-0 dengan jarak 18 km, rute kedua yaitu 0-8-5-0 dengan jarak 23 km, rute ketiga yaitu 0-9-10-6-0 dengan jarak 27 km, dan rute keempat yaitu 0-3-2-0 dengan jarak 32 km. Keempat rute memiliki total jarak yaitu 100 km.

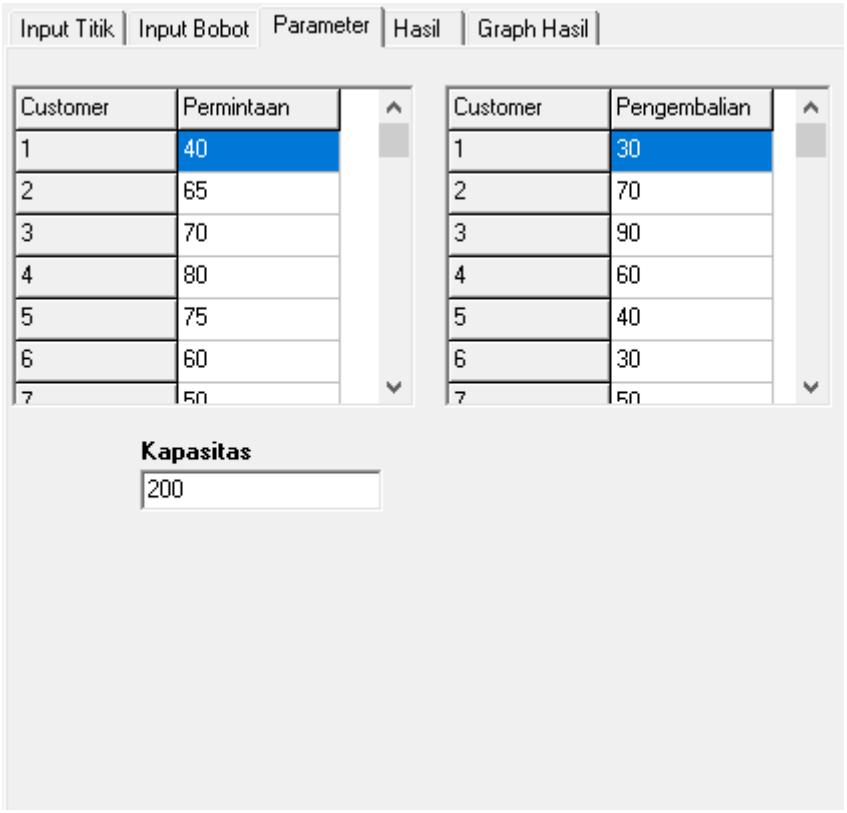


(a)

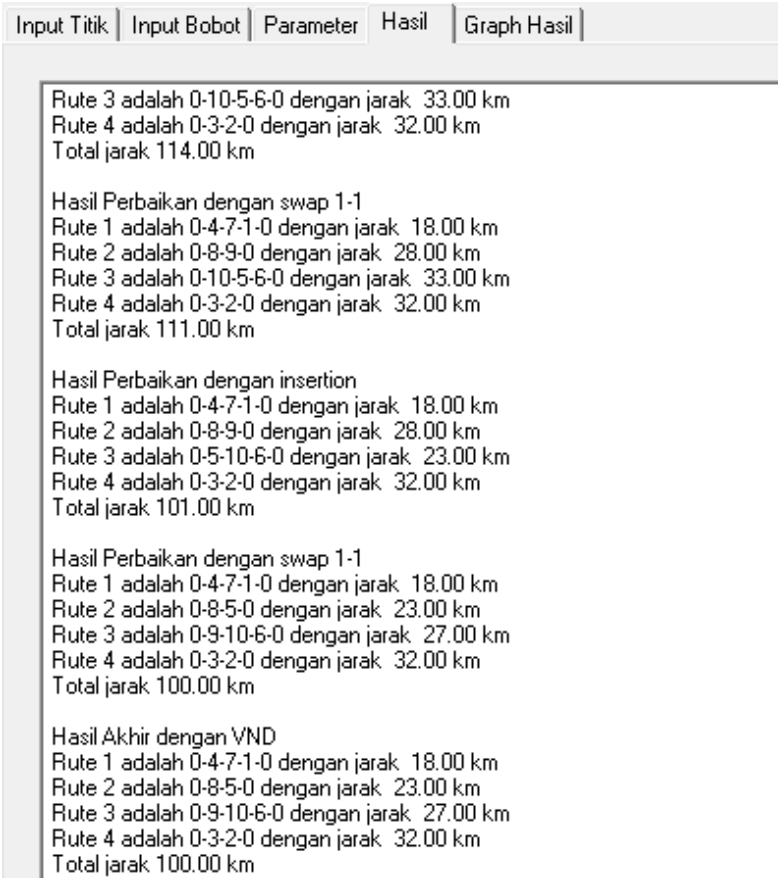
Input Titik | Input Bobot | Parameter | Hasil | Graph Hasil

	0	1	2	3	4	5	6	7	8	9	10
0		3	9	16	4	10	5	11	10	12	15
1	3		6	13	7	7	2	4	11	17	12
2	9	6		7	13	13	8	20	17	22	14
3	16	13	7		20	20	11	17	24	14	7
4	4	7	13	20		13	22	7	6	12	19
5	10	7	13	20	13		8	7	3	9	5
6	5	2	8	11	22	8		15	11	10	3
7	11	4	20	17	7	7	15		4	10	13
8	10	11	17	24	6	3	11	4		6	13
9	12	17	22	14	12	9	10	10	6		7
10	15	12	14	7	19	5	3	13	13	7	

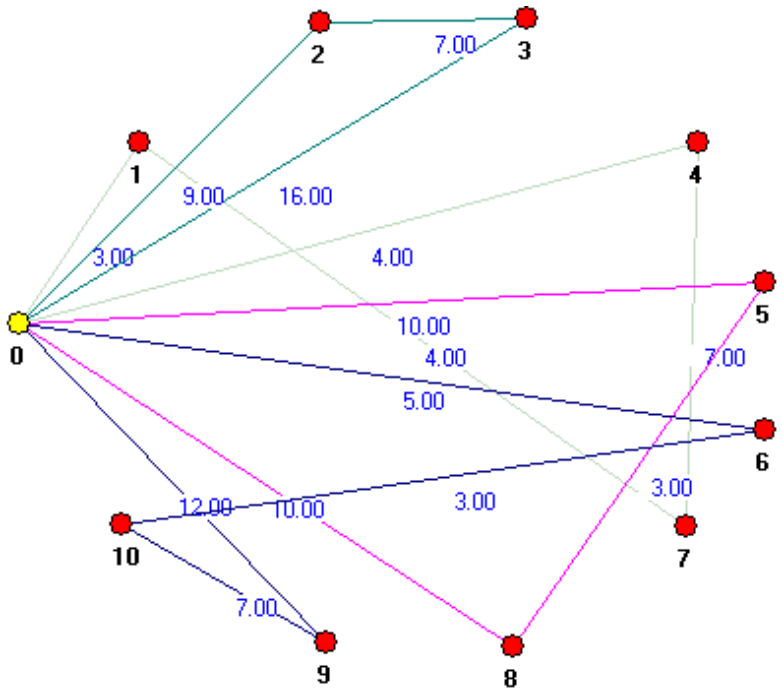
(b)


Input Titik
Input Bobot
Parameter
Hasil
Graph Hasil
Customer
Permintaan
1
40
2
65
3
70
4
80
5
75
6
60
Customer
Pengembalian
1
30
2
70
3
90
4
60
5
40
6
30
Kapasitas
200

(c)


Input Titik
Input Bobot
Parameter
Hasil
Graph Hasil
Rute 3 adalah 0-10-5-6-0 dengan jarak 33.00 km
Rute 4 adalah 0-3-2-0 dengan jarak 32.00 km
Total jarak 114.00 km
Hasil Perbaikan dengan swap 1-1
Rute 1 adalah 0-4-7-1-0 dengan jarak 18.00 km
Rute 2 adalah 0-8-9-0 dengan jarak 28.00 km
Rute 3 adalah 0-10-5-6-0 dengan jarak 33.00 km
Rute 4 adalah 0-3-2-0 dengan jarak 32.00 km
Total jarak 111.00 km
Hasil Perbaikan dengan insertion
Rute 1 adalah 0-4-7-1-0 dengan jarak 18.00 km
Rute 2 adalah 0-8-9-0 dengan jarak 28.00 km
Rute 3 adalah 0-5-10-6-0 dengan jarak 23.00 km
Rute 4 adalah 0-3-2-0 dengan jarak 32.00 km
Total jarak 101.00 km
Hasil Perbaikan dengan swap 1-1
Rute 1 adalah 0-4-7-1-0 dengan jarak 18.00 km
Rute 2 adalah 0-8-5-0 dengan jarak 23.00 km
Rute 3 adalah 0-9-10-6-0 dengan jarak 27.00 km
Rute 4 adalah 0-3-2-0 dengan jarak 32.00 km
Total jarak 100.00 km
Hasil Akhir dengan VND
Rute 1 adalah 0-4-7-1-0 dengan jarak 18.00 km
Rute 2 adalah 0-8-5-0 dengan jarak 23.00 km
Rute 3 adalah 0-9-10-6-0 dengan jarak 27.00 km
Rute 4 adalah 0-3-2-0 dengan jarak 32.00 km
Total jarak 100.00 km

(d)



(e)

Gambar 2. Tampilan Penyelesaian VRPSDP dengan 10 *customer*. (a) *Graph* awal, (b) *Input* Bobot, (c) Input Permintaan, Pengembalian, dan Kapasitas, (d) Hasil akhir yang diperoleh, (e) Rute akhir

PENUTUP

Dalam penyelesaiannya, algoritma VND pada VRPSDP memiliki dua langkah yang digunakan yaitu mencari solusi awal dengan menggunakan algoritma *Insertion Heuristic* dan memperbaiki posisi *customer* dengan menggunakan operator *neighborhood* pada algoritma VND. Perbaikan posisi *customer* dilakukan untuk mencari total jarak minimum tanpa melanggar kendala kapasitas. Dalam melakukan perbaikan posisi *customer*, akan digunakan enam operator *neighborhood*. Dari keenam operator *neighborhood* tersebut, solusi terbaik akan dipilih dan akan dibandingkan dengan solusi awal. Jika semua operator telah dilakukan dan solusi perbaikan yang diperoleh konvergen ke suatu nilai minimum, maka kondisi optimum dari algoritma VND telah tercapai. Implementasi algoritma VND pada VRPSDP telah dibuat dengan menggunakan bahasa pemrograman *Borland Delphi 7*. *Input* yang terdapat pada program adalah posisi titik, jarak antar titik, permintaan dan pengembalian *customer* serta kapasitas kendaraan. *Output* yang terdapat pada program berupa rute-rute yang telah diselesaikan dengan menggunakan algoritma VND pada VRPSDP dan *output* yang berupa gambar dari solusi akhir yang telah diperoleh. Program telah berhasil digunakan untuk menyelesaikan permasalahan VRPSDP dengan 6 *customer* dan 10 *customer*. Berdasarkan hasil yang diperoleh, contoh dengan 6 *customer* menghasilkan 3 rute dengan total jarak sebesar 266 km, sedangkan contoh dengan 10 *customer* menghasilkan 4 rute dengan total jarak sebesar 100 km.

ACKNOWLEDGMENTS

Artikel ini merupakan hasil penelitian hibah skripsi dana PNBP UM, dengan nomor Kontrak: 4.3.474/UN32.14.1/LT/2020, berjudul: “Algoritma Variable Neighborhood Descent (VND) Pada Vehicle Routing Problem With Simultaneous Delivery And Pickup (VRPSDP) dan Implementasinya ”. Terimakasih penulis ucapkan kepada Universitas Negeri Malang atas support dana penelitian ini.

DAFTAR RUJUKAN

Amous, M., Toumi, S., Jarboui, B., & Eddaly, M. 2017. A Variable Neighborhood Search Algorithm for the Capacitated Vehicle Routing Problem. *Electronic Notes in Discrete Mathematics*, 58, 231–238. DOI: 10.1016/j.endm.2017.03.030

Harzi, M. & Krichen, S. 2017. Variable Neighborhood Descent for Solving the Vehicle Routing Problem with Time Windows. *Electronic Notes in Discrete Mathematics*, 58, 175–182. DOI: 10.1016/j.endm.2017.03.023

Herrán, A., Colmenar, J.M., & Duarte, A. 2019. A Variable Neighborhood Search Approach for the Hamiltonian p-median Problem. *Applied Soft Computing Journal*, 80, 603–616. DOI: 10.1016/j.asoc.2019.04.033

Kalayci, C.B. & Kaya, C. 2016. An Ant Colony System Empowered Variable Neighborhood Search Algorithm for the Vehicle Routing Problem with Simultaneous Pickup and Delivery. *Expert Systems with Applications*, 66, 163–175. DOI: 10.1016/j.eswa.2016.09.017

Lai, C., Chen, C., & Ma, Y. 2008. Vehicle Routing Problem with Simultaneously Deliveries and Pickups. *Journal of Far East University*, 25(3), 475–484.

Mu, D., Wang, C., Zhao, F., & Sutherland, J.W. 2016. Solving Vehicle Routing Problem with Simultaneous Pickup and Delivery using Parallel Simulated Annealing Algorithm. *International Journal of Shipping and Transport Logistics*, 8(1), 81-106. DOI: 10.1504/IJSTL.2016.073323

Satyananda, D. & Wahyuningsih, S. 2019. VND in CVRP, MDVRP, and VRPTW cases. *Journal of Physics*, 1320, 1-8. DOI: 10.1088/1742-6596/1320/1/012025.

Sayyah, M., Larki, H., & Yousefikhoshbakht, M. 2016. Solving the Vehicle Routing Problem with Simultaneous Pickup and Delivery by an Effective Ant Colony Optimization. *Journal of Industrial Engineering and Management Studies*, 3(1), 15-38.

Simsir, F. & Ekmekci, D. 2019. A Metaheuristic Solution Approach to Capacitied Vehicle Routing and Network Optimization. *Engineering Science and Technology, an International Journal*, 22, 727-735. DOI: 10.1016/j.jestch.2019.01.002.

Zhu, H., Feng, J., & Li, H. 2017. MN_GLS for VRP with Simultaneous Delivery and Pickup. *Journal of Computers*, 28(6), 1-12. DOI: 10.3966/199115992017122806001.