



저작자표시-비영리-변경금지 2.0 대한민국

이용자는 아래의 조건을 따르는 경우에 한하여 자유롭게

- 이 저작물을 복제, 배포, 전송, 전시, 공연 및 방송할 수 있습니다.

다음과 같은 조건을 따라야 합니다:



저작자표시. 귀하는 원저작자를 표시하여야 합니다.



비영리. 귀하는 이 저작물을 영리 목적으로 이용할 수 없습니다.



변경금지. 귀하는 이 저작물을 개작, 변형 또는 가공할 수 없습니다.

- 귀하는, 이 저작물의 재이용이나 배포의 경우, 이 저작물에 적용된 이용허락조건을 명확하게 나타내어야 합니다.
- 저작권자로부터 별도의 허가를 받으면 이러한 조건들은 적용되지 않습니다.

저작권법에 따른 이용자의 권리는 위의 내용에 의하여 영향을 받지 않습니다.

이것은 [이용허락규약\(Legal Code\)](#)을 이해하기 쉽게 요약한 것입니다.

[Disclaimer](#)

A Doctorate Dissertation in Department of Logistics

**De-Cranking Heuristic Method for
Vehicle Routing Problems**

Supervisor **Kim Hwan-Seong**



Department of Logistics

Graduate School of Korea Maritime University

Tran Ngoc Hoang Son

工學博士 學位論文

A Doctorate Dissertation in Department of Logistics

**De-Cranking Heuristic Method for
Vehicle Routing Problems**

指導教授 金 煥 成



2009. 08

韓國海洋大學校 大學院

東北亞物流시스템學科

Tran Ngoc Hoang Son

本 論 文 을 工 學 博 士 學 位 論 文 으 로 認 准 함

委員長 工學博士 申宰榮



委員 工學博士 郭圭錫



委員 工學博士 南奇燦



委員 工學博士 許允守



委員 工學博士 金煥成



2009 年 08 月

韓國海洋大學校 大學院

東北亞物流시스템學科

Tran Ngoc Hoang Son

Acknowledgements

I am deeply indebted to my supervisor Prof. Kim Hwan-Seong who has given freely of his support, time, suggestions, “pressing” and understanding during my doctorate course in department of Logistics at Korea Maritime University. Great patient from him made me moving to complete this book.

I am grateful to Prof. Shin Jae-Yeong, Prof. Kwak Kyu-Seok, Prof. Nam Ki-Chan, department of Logistics; Dr. Hur Yun-Su, Busan Development Institute. They all helped me immeasurably in detailing comments and clarifying the presentation of material in different parts of the dissertation.

Closely express my thanks to Korean and Vietnamese friends for their friendliness and heartiness. Frequently discussion with them has somewhat maintained and balanced my studying and life: Mr. Kim Hyeon-Hui, Mr Oh Ju-Hyeon.

It is warmly that I would like to thank my parents and my younger brother for their constant supports and encouragements.

Korea Maritime University, Busan, Korea

July 7, 2009

Tran Ngoc Hoang Son

De-Cranking Heuristic Method for Vehicle Routing Problems

Tran Ngoc Hoang Son

Department of Logistics

Graduate School of Korea Maritime University, 2009

Abstract

The field of vehicle routing is currently growing rapidly because of many actual applications in truckload and less-than-truckload trucking, courier services, door-to-door services and many other problems that generally hinder the optimization of transportation costs in a logistics network. The rapidly increasing number of customers in such a network has caused problems such as difficulty in cost optimization in terms of getting a global optimum solution in an acceptable time. Fast algorithms are needed to find sufficient solutions in a limited time that can be used for real-time scheduling.

This dissertation will discuss about heuristics to solve the vehicle routing problems (VRP) in static and dynamic contexts. The solutions for VRP can be obtained exact or heuristic ways. Specially, an introduction to De-Cranking heuristic method, which is an effective improvement of the problem solving method to solve the VRP, will be drawn out. The goal is to minimize the

transportation cost for motor-elements fleet such as vehicles, truck, lorry, train, AGV and etc. moving inside warehouses or touring around a planned ordered list of locations; crews, waiters/waitresses in large custom “Phở” restaurant and etc. moving to server guests; data frames moving on networks and etc. by rerouting and re-dispatching at any time occurring new requests, and/or changing overtime traffic condition on the itineraries.

As any heuristic methods for the VRP, a started solution should be initialized before applying any adjustment procedures from the methods. That solution may be taken from random or reasonable methods. Even with random methods need also some reasonable ones to generate out which related to something called Monte Carlo Methods. This dissertation will not investigate to random methods but will introduce some useful and effective heuristic methods (reasonable ones) for the VRP.



Beginning with the nearest neighbor method, which is classical immediate selecting one next location with lowest cost through all solution until shaping a complete route solution, and then, developing an algorithm in which generalizing the string of location involving in checking lowest cost selection with a given length L . It is called nearest L -neighbor method (NLNM). This method is utilized to obtain the first stage route solution.

In second stage, De-Cranking procedure will release the “cranky energy” if any appearing in the partial route of the first-stage solution, but conserving complete routing to all locations, to get a better route which may not be global optimal solution but near or sometime coincide to it.

The dissertation also gives elements, processes and simulation methods such as lexicographic ordering, traversal the multi-branch tree and so on, from which some exact methods and heuristic method could be developed base on dependently or separately, to solve a class of combinatorial optimization problems which VRP is a representative one.

Keywords: De-Cranking Heuristic Method, Nearest L -Neighbor Method, Vehicle Routing Problem, Dynamic Vehicle Routing Problem, Lexicographic Ordering



Contents

| | |
|--|-----|
| Acknowledgements | i |
| Abstract | ii |
| Contents | v |
| Figure Listing | vii |
| Table Listing | x |
| Chapter 1 Introduction | 1 |
| 1.1 Introduction | 1 |
| 1.2 Outline of the Dissertation | 6 |
| Chapter 2 Vehicle Routing Problem: Static and Dynamic | 8 |
| 2.1 Introduction | 8 |
| 2.2 Static VRP | 8 |
| 2.3 Dynamic VRP | 16 |
| 2.3.1 Dynamic of Unknown Customers | 17 |
| 2.3.2 Dynamic of Travel Times | 18 |
| Chapter 3 Solving the VRPs | 21 |
| 3.1 Introduction | 21 |
| 3.2 Set Theory and Order Relation | 22 |
| 3.3 Mapping | 23 |
| 3.4 Permutation and Combination | 25 |
| 3.5 Multi-branch Tree and Traversal | 27 |
| 3.6 Exact Methods | 28 |

| | | |
|------------------|------------------------------------|-----------|
| 3.7 | Heuristics | 28 |
| 3.7.1 | Nearest L -Neighbor Method | 28 |
| 3.7.2 | De-Crossing | 31 |
| 3.7.3 | De-Cranking | 34 |
| Chapter 4 | Simulation | 39 |
| 4.1 | Simulation Conditions | 39 |
| 4.1.1 | Solomon Data | 39 |
| 4.1.2 | Define Conditions | 41 |
| 4.2 | Simulation Results | 42 |
| 4.2.1 | VRP | 43 |
| 4.2.2 | DVRP | 56 |
| 4.3 | Comments and Remarks | 68 |
| Chapter 5 | Conclusions | 71 |
| | Published Papers | 72 |
| | Appendix | 73 |
| a. | Solomon Data | 73 |
| b. | MATLAB Codes | 75 |
| | Bibliography | 83 |



Figure Listing

| | | |
|-----------------|--|----|
| Fig. 1.1 | Developing of heuristic methods | 5 |
| Fig. 2.1 | Vehicle routing problem | 10 |
| Fig. 2.2 | Serving situations: waiting case | 14 |
| Fig. 2.3 | Serving situations: normal case | 15 |
| Fig. 2.4 | Serving situations: delaying case 1 | 16 |
| Fig. 2.5 | Serving situations: delaying case 2 | 16 |
| Fig. 2.6 | γ continuously changing overtime | 20 |
| Fig. 2.7 | γ average value in each link to customer location | 20 |
| Fig. 2.8 | Uniform pdf of γ | 20 |
| Fig. 3.1 | Multi-branch Tree | 27 |
| Fig. 3.2 | Solution of NNM | 29 |
| Fig. 3.3 | Solution of NLNM with L=2 | 29 |
| Fig. 3.4 | Cross in the route from NLNM | 31 |
| Fig. 3.5 | Crossing (V_1, V_2, V_3, V_4) and de-crossing (V_1, V_3, V_2, V_4) | 32 |
| Fig. 3.6 | De-crossing of example in Fig. 3.4 | 33 |
| Fig. 3.7 | Algorithm for de-cranking | 34 |
| Fig. 3.8 | Example of cranking and de-cranking | 37 |
| Fig. 3.9 | Steps of de-cranking for the example | 37 |
| Fig. 4.1 | The arrangement of customer locations and one depot | 40 |
| Fig. 4.2 | The order of customer request along the working day | 40 |
| Fig. 4.3 | Simulation results of NLNM | 43 |

| | | |
|------------------|---|----|
| Fig. 4.4 | Simulation results of NLNM with de-cranking | 44 |
| Fig. 4.5 | VRP without de-cranking of sample R101 with N=100, K=1 and L=1, case 1 | 45 |
| Fig. 4.6 | VRP with de-cranking of sample R101 with N=100, K=1 and L=1, case 1 | 45 |
| Fig. 4.7 | VRP without de-cranking of sample R101 with N=100, K=1 and L=1, case 2 | 46 |
| Fig. 4.8 | VRP with de-cranking of sample R101 with N=100, K=1 and L=1, case 2 | 46 |
| Fig. 4.9 | VRP without de-cranking of sample R101 with N=100, K=1 and L=1, case 3 | 47 |
| Fig. 4.10 | VRP with de-cranking of sample R101 with N=100, K=1 and L=1, case 3 | 47 |
| Fig. 4.11 | VRP with sample R101 with N=100, K=3 and L=1, case 1 | 48 |
| Fig. 4.12 | VRP with de-crossing of sample R101 with N=100, K=3 and L=1, case 1 | 48 |
| Fig. 4.13 | NLNM with de-cranking, case 1 | 51 |
| Fig. 4.14 | NLNM with de-cranking, case 2 | 53 |
| Fig. 4.15 | NLNM with de-cranking, case 3 | 55 |
| Fig. 4.16 | NLNM (L=1) with de-cranking for DVRP, case 1 | 57 |
| Fig. 4.17 | NLNM (L=1) with de-cranking for DVRP, case 2 | 59 |
| Fig. 4.18 | NLNM (L=1) with de-cranking for DVRP, case 3 | 61 |

Fig. 4.19 NLNM (L=2) with de-cranking for DVRP, case 1 63
Fig. 4.20 NLNM (L=2) with de-cranking for DVRP, case 2 65
Fig. 4.21 NLNM (L=2) with de-cranking for DVRP, case 3 67



Table Listing

| | | |
|-------------------|---|----|
| Table 4.1 | Simulation cases | 41 |
| Table 4.2 | Simulation results of VRP | 43 |
| Table 4.3 | Simulation results of VRP with de-cranking | 44 |
| Table 4.4 | Simulation results of VRP with single vehicle | 49 |
| Table 4.5 | Simulation results of VRP with multi vehicles | 49 |
| Table 4.6 | NLNM with de-cranking, case 1 | 50 |
| Table 4.7 | NLNM with de-cranking, case 2 | 52 |
| Table 4.8 | NLNM with de-cranking, case 3 | 54 |
| Table 4.9 | NLNM (L=1) with de-cranking for DVRP, case 1 | 56 |
| Table 4.10 | NLNM (L=1) with de-cranking for DVRP, case 2 | 58 |
| Table 4.11 | NLNM (L=1) with de-cranking for DVRP, case 3 | 60 |
| Table 4.12 | NLNM (L=2) with de-cranking for DVRP, case 1 | 62 |
| Table 4.13 | NLNM (L=2) with de-cranking for DVRP, case 2 | 64 |
| Table 4.14 | NLNM (L=2) with de-cranking for DVRP, case 3 | 66 |

Chapter 1 Introduction

1.1 Introduction

People are living in a dynamic world where the moving and exchanging of goods or commodities are processing every day, every hours, even every seconds from places to places (arranging products in a warehouse), from place to people (delivery services or goods from supermarket, store or post office to customers) and from people to each others.

To get better and faster services, people create vehicles to support moving or exchanging processes. At the first thought, it could serve for individual or personal purposes, and that vehicle serving operations could be simple which may not necessary to plan for working. But when the services provide for many individuals and industry, there is a vital need to plan and schedule the operations, routes of vehicles in constraints such as limited resources, restricted time operating and space. The goal is to minimize the time on goods transition, and as a result, reducing the cost.

In recent three decade, the vehicle routing problem (VRP) and the more practical version of VRP, the dynamic vehicle routing problem (DVRP), have been studied with much interests because of the importance of those in logistics networks, as reviewed in (Gendreau et al, 1996; Ichoua et al., 2003; Haghani and Jung, 2005; Hashimoto et al., 2006; Fabri and Recht, 2006; Hanshar and Ombuki-Berman, 2007). The VRP has had important points in the scheduling of the routes of vehicles that carry materials, goods, products in a logistics network or chain of suppliers – manufacturers – warehouses and distribution centers – customers (Simchi-Levi et al., 2003), such as door-to-door services, courier services, full truckload (FTL) and less-than-truckload (LTL) services, etc. Recent applications of VRP in the fields of container terminals have been applied to the delivery of containers (Shin and Oh, 2008) or to the planning of a real-time location system (RTLS) (Shin et al., 2008), and so on.

With the increasing of applicable and low-cost modern technologies on precise positioning and communication like as the GPS (Global Positioning System), the GIS (Geographical Information System), traffic flow sensors and cellular telephones that make the VRP and DVRP easier to apply to real-life indoor and outdoor applications (see Ghiani et al. (2003); Taniguchi et al. (2004)).

In 21st century, numerous papers and monographs have been researching and developing efficient exact algorithms and heuristics for the VRP and the DVRP as in Larsen (2000); Ichoua et al. (2000); Toth and Vigo (2002); Branke et al. (2005); Hvattum et al. (2006, 2007); The trend have been turning to develop fast and efficient heuristics methods for VRPs.

The planning routes for vehicles can be scheduled off-line for long term business or on-line for tactical business. We can see that more than 80 per cent of the operations of a logistics network are related to vehicle movements, such as vehicles traveling between stages of supply chain and vehicles moving inside each stage of the supply chain. The control of these movements can enormously affect total cost. Consequently, a strategy in giving routes solution in VRPs is very important.

This planning step is similar to control rules applied to a fleet of vehicles, such as actuators, in stages of the supply chain or logistics network, similar to a controlled system. The purpose is to find an optimal control rule that minimizes the operating cost of actuators or minimizes the transportation cost of vehicles in the VRP. The control rules should change over the operating time to react with the changes in environment (events, traffic condition, etc.) and generate suitable decisions that optimize vehicle operation.

However, it has been known that VRPs is NP-hard. Because of its characteristics, VRPs require many techniques in finding an exact solution, especially by heuristic means. Differences in meta-heuristic methods are discussed by Michalewicz (1996); Toth and Vigo (2002); Chitty et al. (2004); Zeimpekis et al. (2005); Montemanni et al. (2005), and Fan et al. (2006).

Generally, the exact methods are used to exploring all solution space by enumerating techniques (ex. brute force, backtracking, branch and bound etc.), see Horowitz and Sahni (1978), whereas the (meta-)heuristic methods are used to

exploring partially solution space by random or/and reasonable rules of selection, see Martello et al. (1987); Ball et al. (1995a, b).

In real applications, the enumerating all the solution space may not practical because of the enormous solutions having to check for feasible. And if the space is multi-dimension, there is an obsession called “curse of dimension” which the examining all the space to find the best solution would take years even with a simple problem. For example, in the vehicle routing problem with just one vehicle touring around N locations and each location just visited one time, the question is which route should be to minimize the time touring all that locations. The exact answer could be gotten if we checked N -factor ($N!$) possible routes of solution space. It may not too hard for an industrious man with $N < 6$ ($6! = 720$), but in real application, number of locations may greater than hundreds in small business, and thousands to millions in large and very large business in which currently strongest sequence computers could not count.

To solve real applications which the exact methods can not give an answer in acceptable time, some heuristic methods have been developed. The progress of developing heuristic method is shown as in Fig. 1.1. The heuristics have been developing in two directions concurrently. These are random methods and reasonable methods. The algorithms in which induced from reasonable statements indicate the level of robust intelligent of the methods. At first level, the heuristics may have involved some contingencies such as random selecting, or producing a random space of population. However, in more robust intelligent systems, the

contingency is replaced by reasonable elements of selecting or/and sampling or/and producing. This dissertation has been going to that trend.

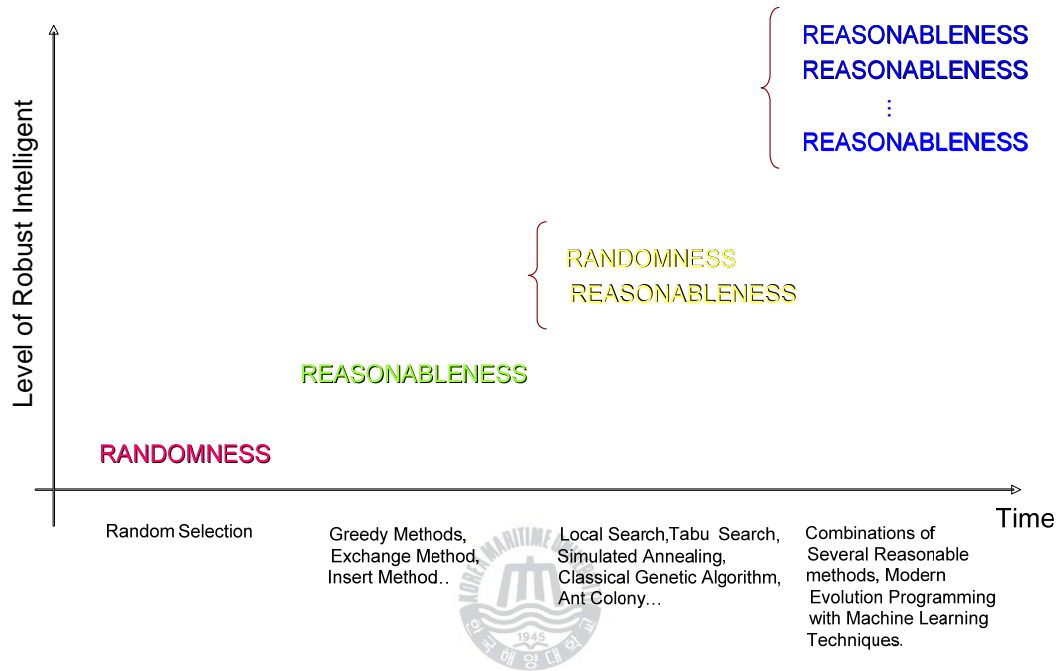


Fig. 1.1 Developing of heuristic methods

For example, producing first solution of the VRP not by random way, but by reasonable fact of nearest L -neighbor method (NLNM) which forming the route by adding continuously a string of length L locations with lowest cost until there is no remain locations. Consequently, the output route from the NLNM is refined by de-cranking procedure, another reasonable one, to produce a better solution.

1.2 Outline of the Dissertation

The purpose of this dissertation will introduce some heuristic methods to solve the VRPs in statics and dynamics context. The nearest L -neighbor method is used to select an initial solution with various values L s at first. And second, the de-branching procedure to “pluck off” the route to decrease cost value of an objective function. Structuring of dissertation is as following.

Chapter 2 will discuss and analyze about the static VRP with time window. The VRP will be formulated mathematically. Analyzing the objective function, detailing several serving cases and costing for each case will be provided clearly. Also, the context of the DVRP will be introduced with degree of dynamics and some modeling of dynamic traffic times.

Chapter 3 will present some mathematical concepts such as set, order relation, mapping, traversal on multi-branch tree for sorting or searching which are helpful to present, manage and solve (programming) the combinatorial optimization problems analytically and computationally. The end of this section will be discussing heuristic methods used to solve the VRPs. Nearest L -neighbor method, simple ideal of de-crossing and philosophy of de-branching procedure will be detailed clearly.

Chapter 4 is simulation results for VRP and DVRP. Testing the nearest L -neighbor method (NLNM) with different values of length L , and enhancing the route solutions with the de-branching procedure to choose the best VRP solutions are presented. The results will show that NLNM with the support of the de-crossing procedure gives a better solution than applying only NLNM, and in many

tests, it even reached an approximate or exact solution. Also, computational results providing an indication of the benefits associated with the de-cranking procedure will be detailed.

Chapter 5 is conclusions about the methods and the obtained results.

The appendix included a sample of Solomon data and all MATLAB codes for heuristics methods presented in the end of chapter 3.



Chapter 2 Vehicle Routing Problem:

Static and Dynamic

2.1 Introduction

VRP is the generalization of the traveling salesman problem (TSP), which is to find the shortest possible tour to make exact single visits to each location. The VRP searches plausible paths or routes from a depot to customer locations for a fleet of capacitated vehicles to serve customers (pick up or/and delivery of goods or commodities) based on optimizing objective functions that indicate benefits (to maximize) or total cost and time (to minimize) of services.

In section 2.2, the mathematical formulations of the static VRP are presented. Section 2.3 discusses some respects of dynamic VRP.

2.2 Static VRP

The basic formulation of objective function for VRP might be referred to Ahuja et al. (1993). Another formulation has been suitable for binary linear

programming referred by Haghani et al. (2005). The former is very general but it does not consider the operational statuses of vehicles. The latter contains too many variables needed to be inputted in the decision solutions, and additionally it is very complex in formulating the problem and not suitable for VRP with a large number of customers. With the consideration of the operational vehicle's status and simple objective function, in this chapter, the following simple objective function will be suggested intuitively.

The vehicle routing problem is formally considered as a complete graph $G = (V, E)$ where $V = \{i \mid i = \overline{0, N}\}$ is the vertexes set and $E = \{(i, j) \mid i \neq j, i = \overline{0, N}, j = \overline{0, N}\}$ is the edges set.

Vertices $i = \overline{1, N}$ are corresponding to customers with N as number of customers, whereas vertex 0 is the depot. A non-negative travel time t_{ij} is associated with the each edge $(i, j) \in E$.

Each customer i is characterized by a pickup location, a service time s_i , a time window $[e_i, l_i]$ and a vehicle planned arrival time t_i . If $t_i < e_i$, the vehicle has to wait up to e_i before servicing the customer and if $t_i > l_i$, the penalty is incurred in the objective.

The depot is characterized by a location, a time window $[e_o, l_o]$ for vehicle arrivals and departures, as well as the vehicle return time t_o^k for each vehicle $k \in K$, where K is the set of vehicles. The service time at the depot is assumed to be

$s_o = 0$. Each vehicle travels along a single route that starts and ends at the depot. The depiction of VRP is shown in Fig. 2.1.

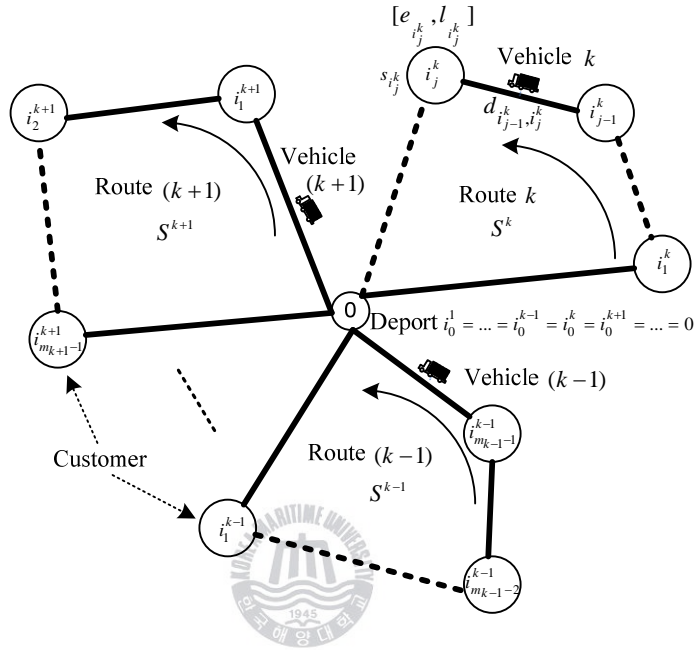


Fig. 2.1 Vehicle routing problem

The notation i_j^k is the customer in j th location the serviced k th vehicle.

$[e_{i_j^k}, l_{i_j^k}]$ is the time window, and $s_{i_j^k}$ is the service time of customer i_j^k in the serving list of the vehicle k . And m_k is the number of customers that vehicle k will have been serving.

The objective of VRP is to minimize the weighted summation of travel time, sum of waiting time at customer locations, sum of delay time at customer

locations and delay time to return to the depot, which formulated in Eq. (2.1) for over all vehicles.

Assume the solution $S = \bigcup_{k \in K} S^k$, where $S^k = \{i_o^k, i_1^k, \dots, i_{m_k}^k\}$ is the sequence of customer locations visited by vehicle k with $i_o^k = i_{m_k}^k = 0$, then the objective function can be expressed as follows:

$$f(S) = \sum_{k \in K} f(S^k) = \sum_{k \in K} \left(\begin{aligned} & a_1 \sum_{j=1}^{m_k} t_{i_{j-1}^k, i_j^k} \\ & + a_2 \sum_{j=1}^{m_k-1} (e_{i_j^k} - t_{i_j^k}^a)^+ \\ & + a_3 \sum_{j=1}^{m_k-1} (t_{i_j^k}^a + s_{i_j^k} - l_{i_j^k}^k)^+ \\ & + a_4 (t_o^k - l_o)^+ \end{aligned} \right) \quad (2.1)$$

where,

a_1, a_2, a_3, a_4 Weighting parameters.

$(*)^+ = \max\{0, *\}$ $(x - y)^+ = \max\{0, x - y\}$

$t_{i_j^k}^a$ Arriving time to customer i_j^k of vehicle k .

$t_{i_j^k}$ Finished time at customer i_j^k of vehicle k and ready to move to next customer.

$$t_{i_{j-1}, i_j}^k = t_{i_j}^a - t_{i_{j-1}}^k$$

$$\sum_{j=1}^{m_k} t_{i_{j-1}, i_j}^k \quad \text{Sum of travel time.}$$

$$\sum_{j=1}^{m_k-1} (e_{i_j}^k - t_{i_j}^a)^+ \quad \text{Sum of waiting time at customer locations.}$$

$$\sum_{j=1}^{m_k-1} (t_{i_j}^a + s_{i_j}^k - l_{i_j}^k)^+ \quad \text{Sum of delay time at customer locations.}$$

$$(t_o^k - l_o)^+ \quad \text{Delay time to come back}$$

If there is no waiting cost of vehicles to depot, then $(e_{i_{m_k}}^k - t_{i_{m_k}}^k)^+ = 0$ or $e_{i_{m_k}}^k \leq t_{i_{m_k}}^k$. Also the ready time in time windows at depot for all vehicles will be zero ($e_{i_{m_k}}^k = 0$), and the travel time of each vehicle is always non-negative value ($t_{i_{m_k}}^k \geq 0$).

Moreover, if the waiting cost of vehicles on their return to depot and on servicing are identical ($a_4 = a_3$), then the objective function can be reduced as follows:

$$\begin{aligned}
f(S) &= \sum_{k \in K} f(S^k) \\
&= \sum_{k \in K} \sum_{j=1}^{m_k} g(S_j^k) \\
&= \sum_{k \in K} \sum_{j=1}^{m_k} \left(\begin{aligned} &a_1 t_{i_{j-1}, i_j^k} \\ &+ a_2 (e_{i_j^k} - t_{i_j^k}^a)^+ \\ &+ a_3 (t_{i_j^k}^a + s_{i_j^k} - l_{i_j^k})^+ \end{aligned} \right) \tag{2.2}
\end{aligned}$$

There are three serving situations at a customer of each vehicle that makes the change in the cost: waiting, normal and delay cases. A detailed objective function of each case will be expressed based on Eq. (2.2) as follows:



a) Waiting case

When the vehicle arrives at the customer location before the ready time (the time the customer needs to be served), then the vehicle has to wait for the right time (the customer accepts the receiving service), as shown in Fig. 2.2. The real total time serving this customer, $T_{i_{j-1}, i_j^k}^k$, and the cost for this link, $g(S_j^k)$, in this case will be given as follows:

$$\begin{aligned}
T_{i_{j-1}, i_j^k}^k &= t_{i_j^k} - t_{i_{j-1}}^k \\
&= e_{i_j^k} - t_{i_{j-1}}^k + s_{i_j^k} \tag{2.3}
\end{aligned}$$

$$g(S_j^k) = a_1 t_{i_{j-1}^k, i_j^k} + a_2 (e_{i_j^k} - t_{i_j^k}^a) \quad (2.4)$$

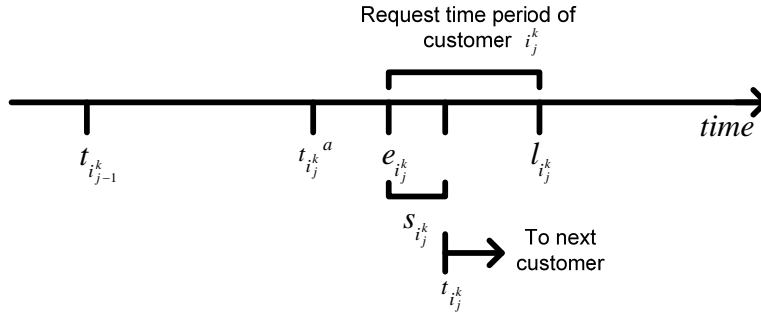


Fig. 2.2 Serving situations: waiting case

b) Normal case

Normal case or the right time case is shown in Fig. 2.3. There is no penalty in the cost link. The real total spent servicing time of the customer i_j^k and the cost function for this link will be calculated as follows:

$$\begin{aligned} T_{i_{j-1}^k, i_j^k} &= t_{i_j^k} - t_{i_{j-1}^k} \\ &= t_{i_j^k}^a - t_{i_{j-1}^k} + s_{i_j^k} \end{aligned} \quad (2.5)$$

$$g(S_j^k) = a_1 t_{i_{j-1}^k, i_j^k} \quad (2.6)$$

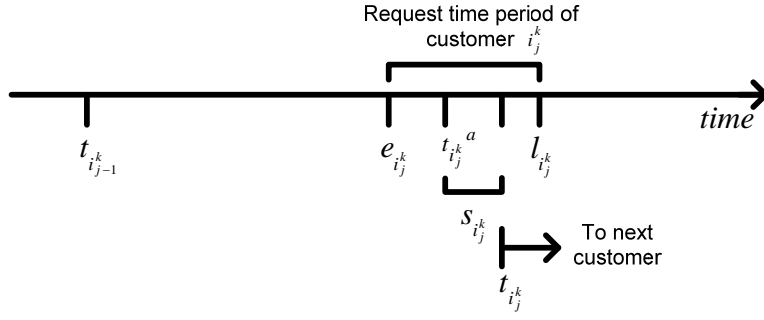


Fig. 2.3 Serving situations: normal case

c) Delay cases

Slowing cases are showed in Figs. 2.4 and 2.5, respectively. In Fig. 2.4, the vehicle arrives in a valid period time, but the servicing time exceeds the due time. In Fig. 2.5, the vehicle reaches the customer totally late. There is a penalty for this lateness. The real total time servicing this customer, T_{i_{j-1}, i_j}^k , and the cost for this link, $g(S_j^k)$, in this case will be calculated as follows:

$$\begin{aligned}
 T_{i_{j-1}, i_j}^k &= t_{i_j}^k - t_{i_{j-1}}^k \\
 &= t_{i_j}^a - t_{i_{j-1}}^k + s_{i_j}^k
 \end{aligned} \tag{2.7}$$

$$g(S_j^k) = a_1 t_{i_{j-1}, i_j}^k + a_3 (t_{i_j}^a + s_{i_j}^k - l_{i_j}^k) \tag{2.8}$$

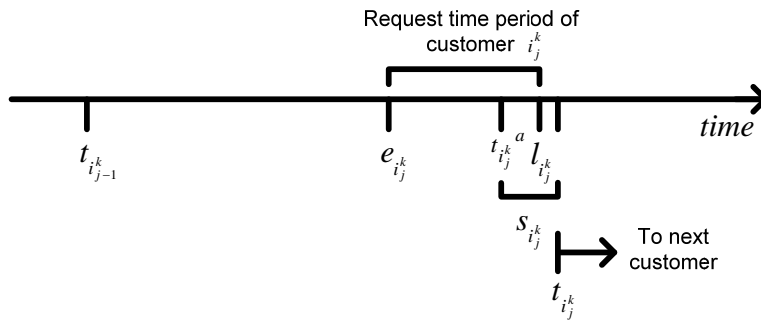


Fig. 2.4 Serving situations: delaying case 1

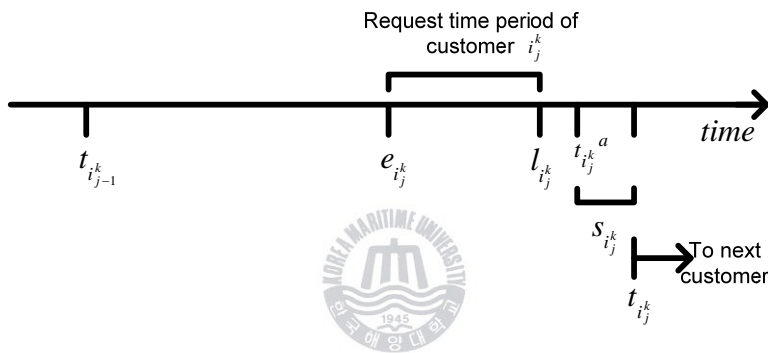


Fig. 2.5 Serving situations: delaying case 2

2.3 Dynamic VRP

Dynamic vehicle routing problem are defined base on static vehicle routing problem adding with a number of changing elements during operating time of vehicles fleet in which a rearrangement all or an adjustment partially of the current routes in real time to adapt with the changing. The changing elements could be previously unknown such as stochastic customers whose requests and locations randomly occur at anytime, dynamics of travel time because of unknown

traffic condition affecting the velocity moving of fleet, and dynamics of vehicle dispatching, or the changing of policies online.

In this section, we will discuss about dynamics of unknown customers and dynamics of travel times.

2.3.1 Dynamics of Unknown Customers

The VRP mentioned in previous chapter is static because of its input data (travelling time, total demand) are known in advance, all data are known when designing vehicles routes. The DVRP are different in different ways as discussed in Hvattum et al. (2006).

It is significant to recall the dynamic degree of DVRP in Larsen (2000). Assume that the planning interval is $[0, T]$. Let n_s and n_d be the number of static and dynamic requests, respectively. Let $t_i \in [0, T]$ be the occurrence time of service request i with earliest time e_i and latest time l_i . Larsenian degree of dynamics is defined as:

$$\delta = \frac{\sum_{i=1}^{n_s+n_d} \frac{t_i + (T - l_i)}{T}}{n_s + n_d}$$

If the latest times l_i are far from the finished working time T , this case is strongly dynamics, whereas if l_i s near T , it is weakly dynamics.

This dissertation will study and simulate the dynamics of customers with various degrees. The dynamic degree of customers in this dissertation is the number of customer requests per a period of rerouting during the operations of vehicles.

The nearest L-neighbor method is used to predict the routes for each of vehicles consequently, and applying de-cranking procedure to reduce the cost before selecting the next customers to serve. One customer is chosen for each vehicle to serve next until there is no more the changing of the inputs.

2.3.2 Dynamics of Travel Times

When the vehicle is ready to depart from its current customer location, the travel time to its next destination is sum of scheduled average time, and variable time amount. This changeable time amount may positive (e.g. high traffic density, accident, bad conditions or downgrading on road) or negative (e.g. low traffic density, good conditions on road) change might be due to unforeseen events that may occur along the current travel section and represents the truly dynamic component of the travel time. Totally, the travel time is fluctuated and it is considered a rescheduling of the planned routes. We model dynamic travel time as follows:

$$t_{i_{j-1}^k, i_j^k} = \frac{d_{i_{j-1}^k, i_j^k}}{v_a} (1 + \gamma) \quad (2.9)$$

where:

- d_{i_{j-1},i_j} Distance between two customer locations.
- v_a General velocity of vehicles, planned speed.
- γ Unknown stochastic coefficient related to traffic condition.
- $\gamma_o \leq \gamma < \infty$,
- γ_o Goodness factor of traffic condition.
- $-1 < \gamma_o \leq 0$,

Some special cases:

- $\gamma \rightarrow -1$ Best traffic condition could not support the vehicle go at infinite speed, so this is impossible case.
- $\gamma = 0$ Normal traffic condition as planned.
- $\gamma \rightarrow \infty$ Serious traffic, so the vehicle can not move.
- $\gamma = -0.5$ Vehicle runs at double planned speed.

γ is related mainly to traffic density and fluctuated during day time. It changes continuously as in Fig 2.6. For testing purposes we can set an identical average value for each link that the vehicle running, in Fig. 2.7. And we assume the distribution of γ is uniform. The value of γ spans from $[-0.5, 8]$ and has uniform probability distribution function $pdf(\gamma)$ as in Fig. 2.8.

This modeling of travelling time would significant because it transforms

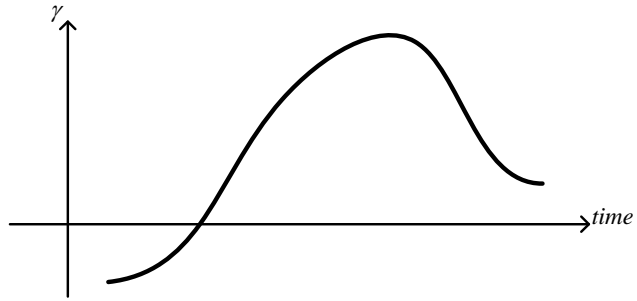


Fig. 2.6 γ continuously changing overtime



Fig. 2.7 γ average value in each link to customer location

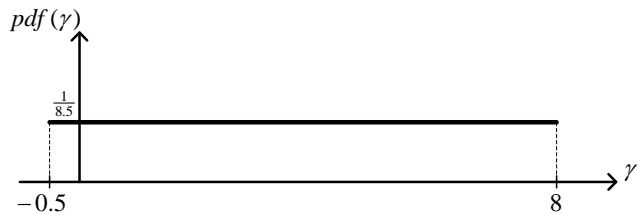


Fig. 2.8 Uniform pdf of γ

the local travelling time t_{i_{j-1}, i_j}^k to a realistic global statistical measurable quantity

γ which is useful in prediction purposes.

Chapter 3 Solving the VRPs

3.1 Introduction

To solve the VRPs on computer, we need some knowledge to present elements of solution space and relation to assess or elements to be arranged on which is important to define a method to search an elements in general space. That is about set theory, order relation, mapping, combination sets, permutation sets, or tree concept and method to examine all tree branches to get exact solutions.

But in real problems, the exploring totally the space solutions is impractical, it may need months, years, or even generations to search real optimum solution for a problem need to be decided in bound time. A fast problem solving ideal may not obtain a best one, but currently solve the problems in a simple way quickly and satisfy the need in acceptable time is a right choice. We call that be heuristic methods.

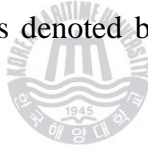
Some of the basic mathematical concepts with exact methods and heuristics methods will be discussed over promptly and end with detailed

introduction to three heuristic methods: Nearest L-Neighbor Method, De-crossing and De-cranking.

3.2 Set Theory and Order Relation

A set is a collection of objects. The objects in a set are also called the elements, or members of the set. We use braces $\{\}$ to indicate a set and parentheses $(,)$ to indicate an ordered set or sequence. A sequence of indexing locations from which constructs the route of vehicle $S^k = \{i_o^k, i_1^k, \dots, i_{m_k}^k\}$ is an example closely to the subject of VRP.

A subset of the set S is a set S^k such that every element of S^k is also a member of S . This relationship is denoted by $(s \in S^k) \Rightarrow (s \in S)$; or $S^k \subset S$; or $S \supset S^k$.



Union of S^1 and S^2 is denoted as $S^1 \cup S^2$. The union of the sets S^1, S^2, \dots, S^K , denoted in short by $\bigcup_{k=1}^K S^k$.

Intersection of S^1 and S^2 is denoted by $S^1 - S^2$ or $S^1 \setminus S^2$.

The Cartesian product of the sets S^1, S^2, \dots, S^n , denoted by $S^1 \times S^2 \times \dots \times S^n$, is the set of n -tuples (s_1, s_2, \dots, s_n) , where s_i belongs to S^i for $i = 1, 2, \dots, n$. We can write it as $S^1 \times S^2 \times \dots \times S^n = \{(s_1, s_2, \dots, s_n) | s_i \in S_i, i = \overline{1, n}\}$.

If the sets S^1, S^2, \dots, S^n are all equal to a set S , then we have n -fold Cartesian product of S which is denoted by $S^1 \times S^2 \times \dots \times S^n = S^{(n)}$.

Lexicographic ordering " \prec " of $S^{(n)}$ is a set of all n -tuples of elements of S , in which $S^{(n)}$ is the n -fold Cartesian product of S . Let $s = (s_1, s_2, \dots, s_n)$ and $t = (t_1, t_2, \dots, t_n)$, $s, t \in S^{(n)}$. Then $s \prec t$ if either $s = t$ or there is some k , $1 \leq k \leq n$, such that $s_i = t_i$, $i = \overline{1, k-1}$ and $s_k < t_k$.

Example 3.1

$$S = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$$

$$n = 9$$

$$s = (1, 2, 3, 6, 5, 4, 7, 8, 9)$$



$$k = 4$$



$$t = (1, 2, 3, 7, 8, 9, 4, 5, 6)$$



At $k = 4$, $s_4 = 6 < t_4 = 7$, and $s_i = t_i, i = \overline{1, 3}$, so that $s \prec t$

3.3 Mapping

Let S and T be given sets. A function $f : S \rightarrow T$ consists of two sets S and T together with a rule that assigns to each $s \in S$ a specific element of T ,

denoted $f(s)$, or s is mapped to the element $f(s) \in T$. We say that S is the domain of f , and T is the co-domain of f .

Function f is said to be injective (one-to-one) if and only if $f(s_1) = f(s_2)$ implies that $s_1 = s_2$ for all s_1 and s_2 in the domain of f .

Function f is surjective (onto) if and only if for every element of S , $s \in S$, there is an element of T , $t \in T$, with $f(s) = t$.

Function f is bijective (one-to-one correspondent), if it is both injective and surjective.

Example 3.2

A distance function on the set of locations S on a axis as

$$\delta : S \times S \mapsto R$$

$$\delta(s_1, s_2) = |s_1 - s_2|$$



Or mapping a lexicographical ordering to integer set as

$$\prec : S^n \mapsto N$$

This mapping is very useful in searching solution of combinatorial problem. It maps n-dimensional space to well-order-one-dimensional structure of integer field \mathbf{N} , so it is easier to move and count through the domain solution space.

3.4 Permutation and Combination

A permutation of a set distinct object is an ordered arrangement of these objects.

Given N objects distinctly, number of arrangement is:

$$P(N) = 1 \times 2 \times 3 \times \dots \times (N-1) \times N = N!$$

$$P(0) = 1 = 0!$$

A k -combination of elements of a set is an unordered selection of k element from the set, then the number of k -combination is:

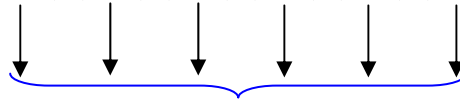
$$C_k^N = \frac{P(N)}{P(k) \times P(N-k)}$$



Example 3.3

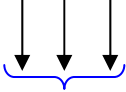
$$S = \{1, 2, 3\}$$

$$\text{Permutation set } P_S = \{(1, 2, 3), (1, 3, 2), (2, 1, 3), (2, 3, 1), (3, 1, 2), (3, 2, 1)\}$$



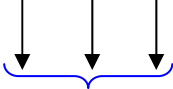
$$P(3) = 1 \times 2 \times 3 = 3! = 6 \text{ elements}$$

1-Combination set $C_S \binom{3}{1} = \{\{1\}, \{2\}, \{3\}\}$




$$C_1^3 = \frac{P(3)}{P(1) \times P(2)} = 3 \text{ elements}$$

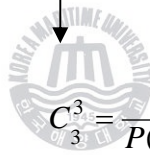
2-Combination set $C_S \binom{3}{2} = \{\{1,2\}, \{2,3\}, \{1,3\}\}$



$$C_2^3 = \frac{P(3)}{P(2) \times P(1)} = 3 \text{ elements}$$

3-Combination set $C_S \binom{3}{3} = \{\{1,2,3\}\}$





$$C_3^3 = \frac{P(3)}{P(3) \times P(0)} = 1 \text{ elements}$$

An algorithm to generate a permutation of set $\{1,2,\dots,n\}$ can be based on a procedure to construct a next order (forward or backward) of a lexicographic order following a given permutation (s_1, s_2, \dots, s_n) . The MATLAB codes to implement next or previous permutation and next k-combination could be referred to appendix.

3.5 Multi-branch Tree and Traversal

Start from a root with multi branches which denote switch decisions. Each branch goes to a node and sniping to decision directions (branching) until an end node which defines a characteristic to stop the branching process. There are two basic methods to search in multi-branch tree, those are depth first search and breadth first search, which is the core of exact methods to solve the VRP. Details of these could be referred to Horowitz et al.(1978) and Russell et al.(2003).

Fig. 3.1 illustrates a example of multi-branch tree. Start node is the root, it is assigned with value from an evaluating function of current state. Finished node

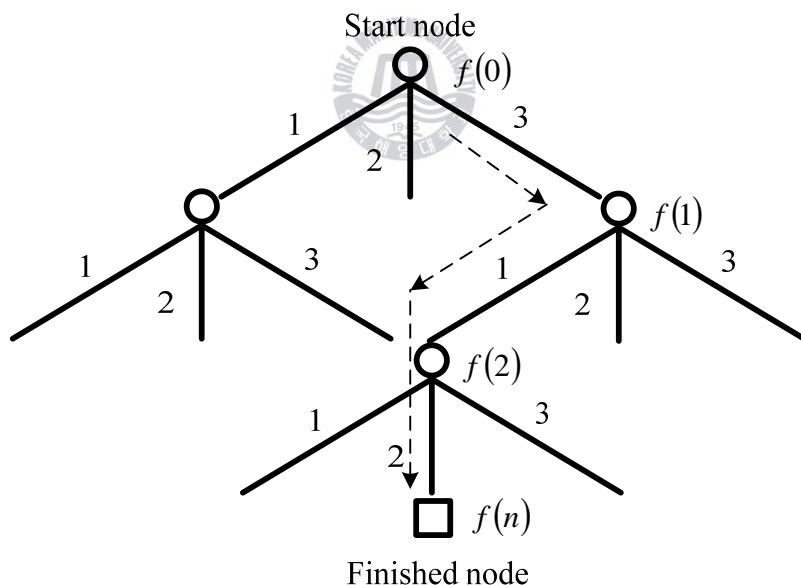


Fig. 3.1 Multi-branch Tree

is the node which the state satisfies every conditions of a problem. The dot lines with arrow indicate the route for a solution.

3.6 Exact Methods

Backtracking, branch and bound, branch and cut, set-covering integer programming are the kinds of called exact methods. “Exact” that means there is a mechanism that visits all the significant elements of searching space. A smart exact method is the one can eliminate unnecessary elements, and just relevant ones to confirm feasibility and use that information to trim others. Details of back tracking algorithm and branch and bound could be refer to Horowitz et al.(1978). The branch and cut and set-covering integer programming and many others could be referred to Toth et al.(2002).

The next section will present nearest L-neighbor method and from de-crossing to de-cranking procedure that be useful to get the solutions in vehicle routing problems. Other Heuristics could be referred to Ball et al.(1995a,b) and Michalewicz et al.(2004)

3.7 Heuristics

3.7.1 Nearest *L*-Neighbor Method

The nearest neighbor method (NNM) is known as a technique for finding the closest point in metric spaces (Arya et al., 1994). A generalization of the nearest neighbor method (NNM) is used to plan vehicular routes. In the nearest

neighbor method, at the current step, only one nearest customer location in all remaining un-served customers is chosen, whereas in nearest L -neighbor method (NLNM), a sequence of L customers is selected to ensure the least total cost.

As an example, Figs. 3.2 and 3.3 show the solutions of NNM and NLNM with $L=2$, respectively, for five customers located in Euclidean space. The solution in Fig. 3.3 looks smoother and shorter than that in Fig. 3.2.

Therefore, NNM is a special case of NLNM when $L=1$ along selecting

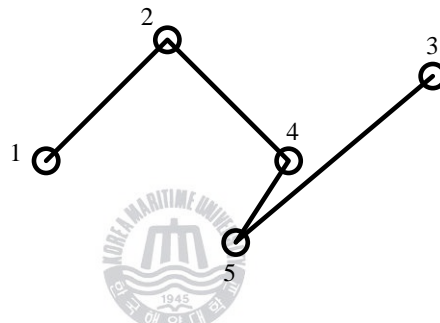


Fig. 3.2 Solution of NNM

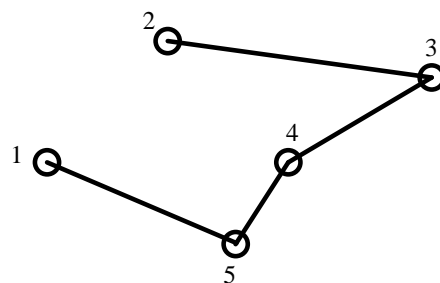


Fig. 3.3 Solution of NLNM with $L=2$

progress. When L equals to the total number of customers, then NLNM gives an exact solution. However, when the number of customers is sufficiently large as in real applications, the time to get the exact solution is unacceptable. At a time we plan a solution for a group of customers, and continuously we plan for other groups until all the locations are considered.

For a graph of N vertexes, NNM needs $C = N(N+1)/2$ comparison steps to get a suboptimum solution.

In case of NLNM, it needs:

$$C = \frac{N!}{(N-L)!} + \frac{(N-L)!}{(N-2L)!} + \dots + \frac{(N-(n-1)L)!}{(N-nL)!} + (N-nL)! \quad (3.1)$$

$$n = \left\lfloor \frac{N}{L} \right\rfloor \quad (3.2)$$

where $M! = M \times (M-1) \times \dots \times 2 \times 1$ and $\lfloor * \rfloor$ means the greatest integer in the argument.

In Eq. (3.1), let $L \rightarrow N$, then $C \rightarrow N!$. This is the total number of elements in solution space. Theoretically, we could check all the elements of this space to get the exact solution.

Actually, when L reaches all existent requests N at the first step and if N is sufficiently large (>30), it might take much computational time to sort out from $N!$ possible permutations in order to obtain the best route solution (exact solution).

Let us give a set V customers and define that $A^L(V)$ is all sets of L customers in set V , and ${}^u S^k$ is set of customers served or will be served of vehicle k at time u by NLNM. Then, we have:

$$A_{\min}^L(V) = \arg \left\{ \min_{A^L(V)} [f(A^L(V))] \right\} \quad (3.3)$$

$${}^u S^k = {}^{(u-1)} S^k \cup A_{\min}^L(V - S) \quad (3.4)$$

Eqs. (3.3) and (3.4) are useful in updating the routes in static VRP and DVRP.

3.7.2 De-Crossing

NLNM sometimes gives a solution with crosses, as shown for example in

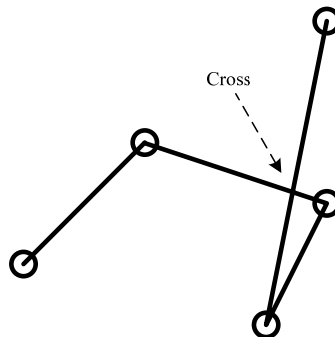


Fig. 3.4 Cross in the route from NLNM

Fig. 3.4 which makes the route look like a bad solution. The de-crossing procedure in the next subsection is to remove the crosses and reduce the length of the route. Therefore, it enhances the quality of the solution.

Assume a crossing situation as a part of a solution called $S_{cross} = (V_1, V_2, V_3, V_4)$ which is shown in Fig. 3.5.

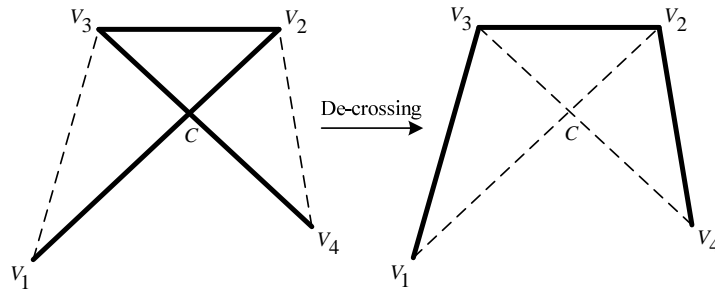


Fig. 3.5 Crossing (V_1, V_2, V_3, V_4) and de-crossing (V_1, V_3, V_2, V_4)

The intersect point is $(V_1, V_2) \cap (V_3, V_4) = \{C\}$.

In Fig. 3.5, the following inequality equations are induced

$$d_{V_1, C} + d_{C, V_3} \geq d_{V_1, V_3} \quad (3.5)$$

$$d_{V_2, C} + d_{C, V_4} \geq d_{V_2, V_4} \quad (3.6)$$

By adding Eq. (3.5) and Eq. (3.6), we have

$$d_{V_1, C} + d_{C, V_3} + d_{V_2, C} + d_{C, V_4} \geq d_{V_1, V_3} + d_{V_2, V_4} \quad (3.7)$$

Also by reducing Eq. (3.7), Eq. (3.8) will be obtained

$$d_{V_1, V_2, V_3, V_4} \geq d_{V_1, V_3, V_2, V_4} \quad (3.8)$$

Inequality Eq. (3.8) implies that the changing order of (V_1, V_2, V_3, V_4) to (V_1, V_3, V_2, V_4) by swapping (or de-crossing as in Fig. 3.6) $\{V_2\}$ and $\{V_3\}$ that reduces the distance from $\{V_1\}$ to $\{V_4\}$, and consequently, it reduces cost. That is the effect of the de-crossing procedure.

By applying the de-crossing procedure in Fig. 3.4, we have a de-crossing result route as shown in Fig. 3.6.

In general, the d_{V_1, V_x, V_y, V_4} in inequality Eq. (3.8) is not only the distance from $\{V_1\}$ to $\{V_4\}$ in Euclidian space but also is the value from the objective function to the set (V_1, V_x, V_y, V_4) . The continuous exchange or release cranky

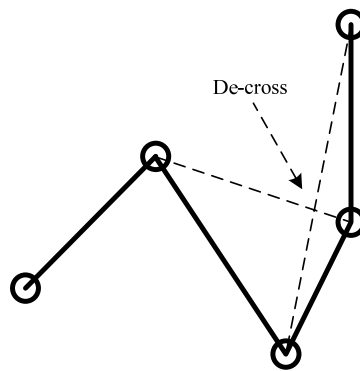


Fig. 3.6 De-crossing of example in Fig. 3.4

energy of partial route to all the route is called de-cranking procedure.

3.7.3 De-Cranking

The ideal of de-cranking based on the act of straightening a bundle messy long wire. At first, two hands hold at one side of the wire. Second, slide one hand along the wire while the second hand keeps and stretches the line. This action will make the wire straight and release crosses. Third, reach gradually the second hand to the first hand position and do same actions on second step. Forth, repeat from first to third step until is straight satisfactorily. That is called de-cranking process, and the “stretching” operation is equivalent to exchange a partial route. We apply this procedure to exist routes which generated from nearest L-neighbor method.

The algorithm in Fig. 3.7 could be presented as pseudo-code below:

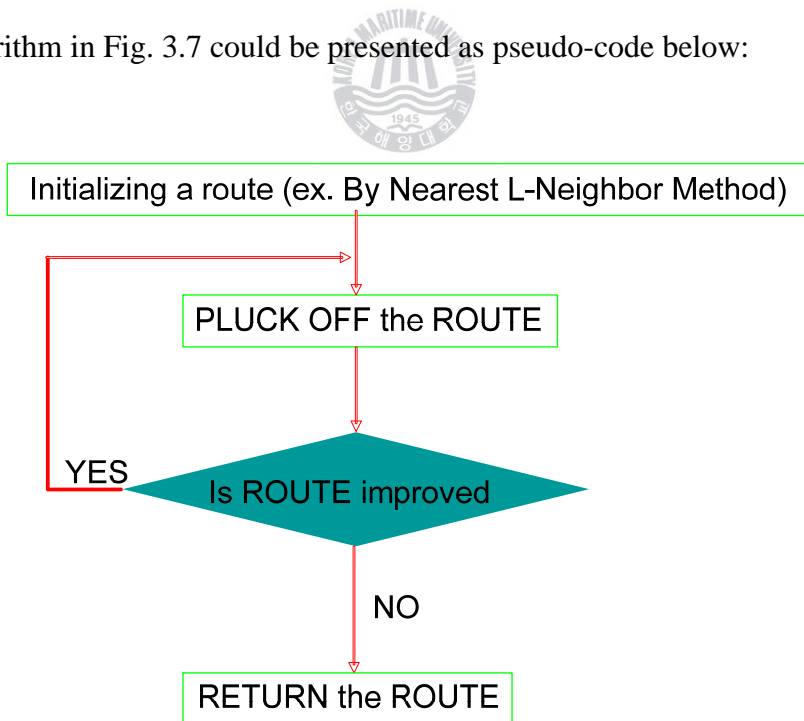


Fig. 3.7 Algorithm for de-cranking


```

Begin_Function New_route = DE_CRANKING( Old_route )

New_route = Old_route;

L = LENGTH( Old_route );

DO{

    Old_route = New_route;


    FOR u = 2 TO L-1 {

        FOR v = u+1 TO L-1 {

            Temp_route = New_route;

            s = u; t = v;

            WHILE( s < t ) {

                 SWAP( Temp_route, s, t );

                s = s + 1; t = t - 1;

            }

            IF( COST( Temp_route ) < COST( New_route ) )

                THEN New_route = Temp_route;

        }

    }

} STOP IF( New_route == Old_route ) ;

RETURN New_route;

End_Function DE_CRANKING

```

As we see in the algorithm, the cost will go down every loop until no more reducing, and the route is converged. The **COST** function can be seen as potential energy of the route state, and the **SWAP** will release the “cranky energy” if it exists in the route state.

The “cranky” is defined in Merriam-Webster’s 11th Collegiate Dictionary as “full of twists and turns”. The “cranky energy” is a kind of potential energy which appearing in an un-optimal route with twist-parts (crossing) and turn-parts along the route which can be detected by a metric function as the objective function. The trick to check twist-parts or/and turn-parts is just a comparison the routes before and after exchange the nodes in systematical way that we can see in the pseudo-algorithm, or in previous explanation of the idea.

Let give a simple example to clarify the ideal of de-cranking energy. A partial route with twist and turn inside, after de-cranking, the energy of twist and turn was cleared away and give a better route as in Figs. 3.8 and 3.9.

The objective function is kind of evaluated function to the energy state of solution set. And as nature of a existing object (biologic-ware or non-biologic-ware such as people, animal, or things, or routes as the object to this dissertation), it should adjust its state energy to the lowest and match with the level “energy of environment” around (living place, political status, supported facilities, relationships, or position of customer locations to which the dissertation concerned as an example), because in this level of energy it can reduce waste energy to against unwanted forces generated by the difference of energies.

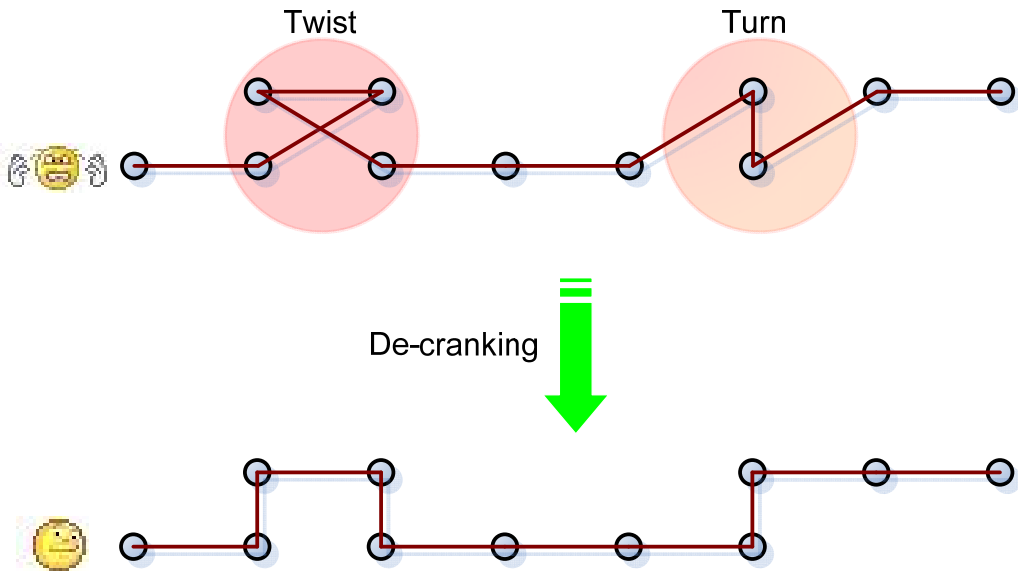


Fig. 3.8 Example of cranking and de-cranking

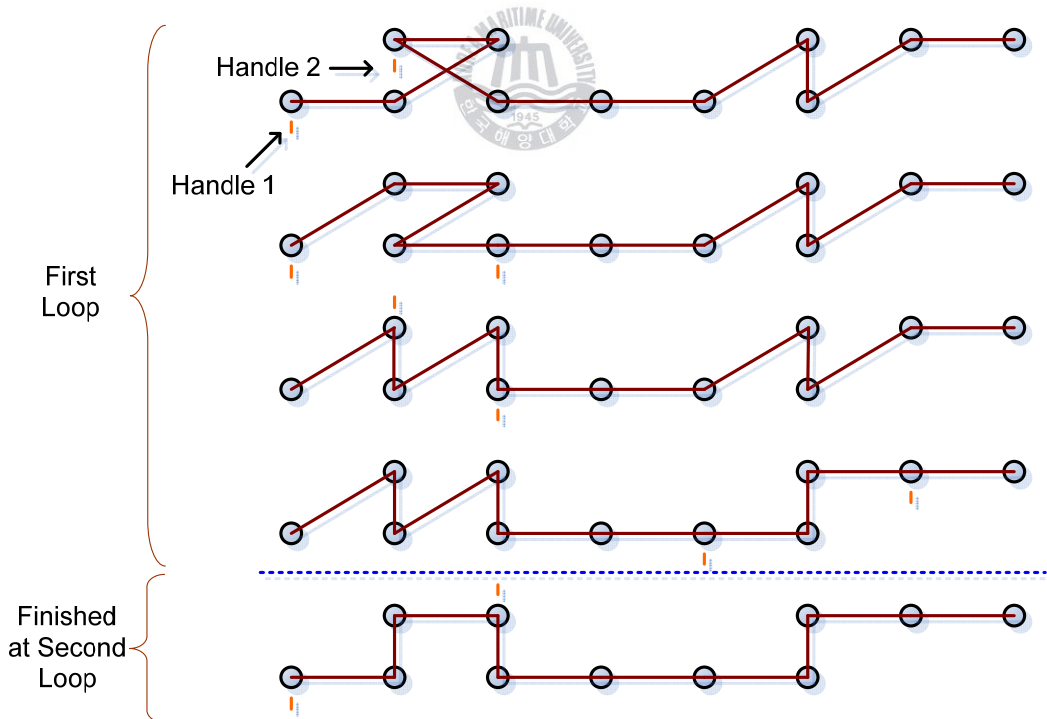


Fig. 3.9 Steps of de-cranking for the example

The de-cranking procedure will release the stagnancy of waste energies (waste costs) at twisting and turning parts along the route, so that its results will set the route with a energy no greater than that route before do de-cranking, that mean is reducing the cost.



Chapter 4 Simulation

4.1 Simulation Conditions

The simulation results will be presented separately the VRP and DVRP. But firstly, we mention about the data to use in simulation study, and preparing data to simulate.



4.1.1 Solomon data

To present the results and to show the effectiveness of the NLNN and de-cranking procedure to the VRP and DVRP, we tested the algorithms on samples of Solomon data. One sample includes 100 customers with a different spatial distribution of location are selected, where its schedule for serving includes ready time, due date, service time, and capacities are considered. Also, Solomon data have three kind of samples R, C, and RC.

R(random) sample distributes the customer location randomly; C (clustered) sample clusters the customer location in well-defined geographic

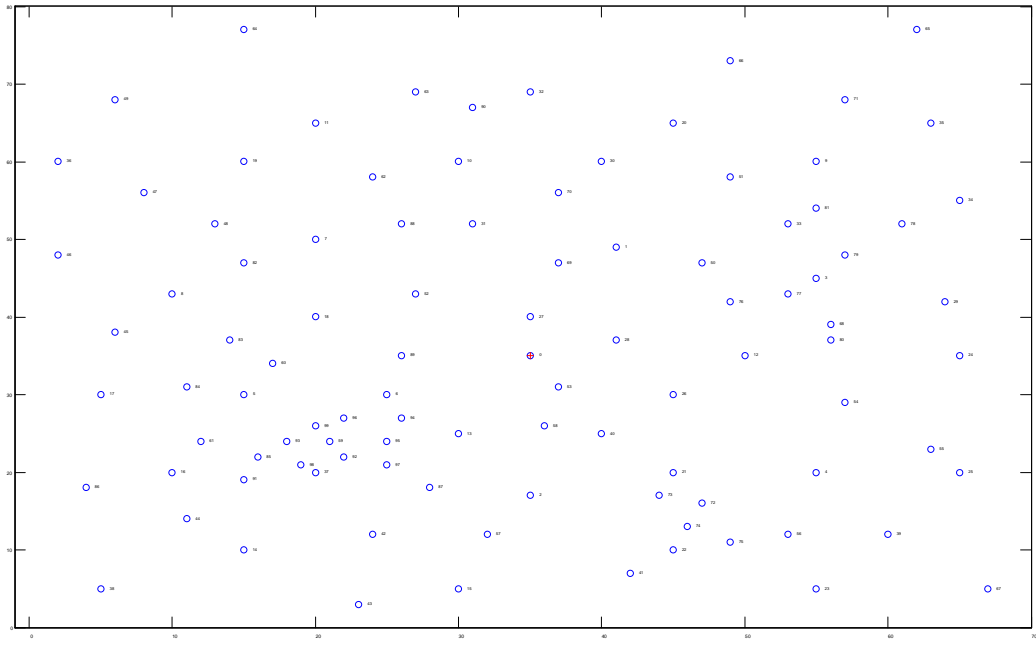


Fig. 4.1 The arrangement of customer locations and one depot

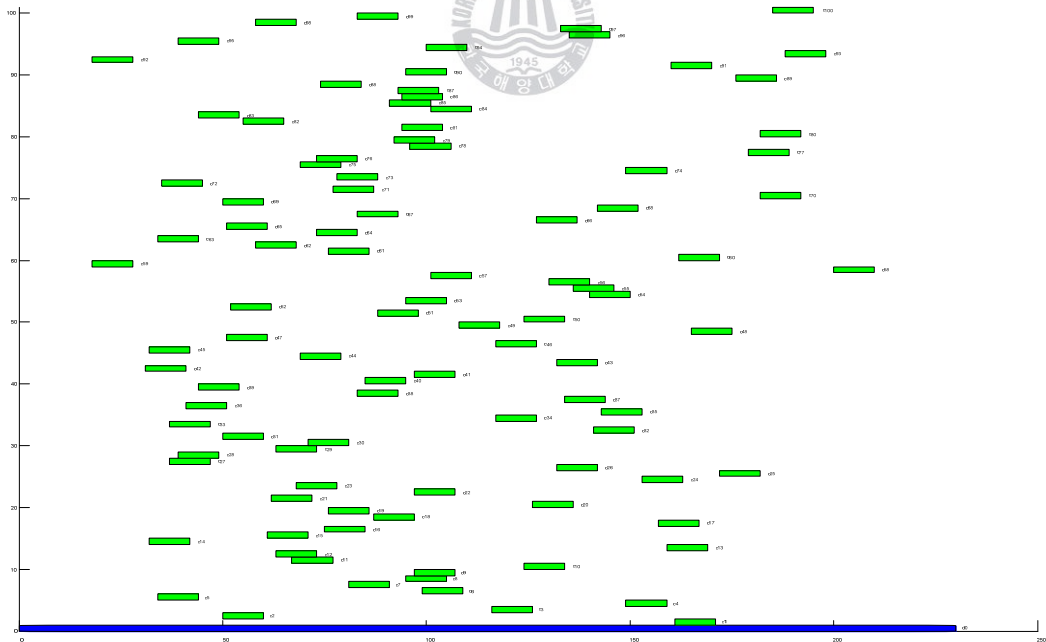


Fig. 4.2 The order of customer request along the working day

cluster; RC samples consider the customer location of both cases (random and clustered)

One of them is R101 is been using to make intuitive of the results. Fig. 4.1 figures out the random locations in sample R101 which has 100 customer locations and one depot. Fig. 4.2 shows the ordering time window of requests during working time with the long blue box indicates the working time, and small green boxes are interval request time.

4.1.2 Define Conditions

Using MATLAB to program on a Pentium Core i7 CPU with 3GB Ram and get the results.

From the objective function in (2), we divided three cases for simulation as show in Table 4.1.

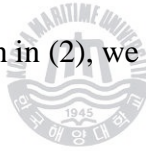


Table 4.1 Simulation cases

| Cases | a_1 | a_2 | a_3 | Comments |
|--------|-------|-------|-------|---|
| Case 1 | 1 | 0 | 0 | Only consider travel times |
| Case 2 | 1 | 0 | 1 | Consider travel times and cost of delay serving |
| Case 3 | 1 | 1 | 1 | Consider travel times and cost of waiting and delay serving |

The L parameter was chosen 1 and 2 to get the simulation for VRP and DVRP.

The maximum number of vehicles involved to simulation is 20 units.

The highlight boxes in Tables from 4.6 to 4.8 indicate the best results from all that cases of simulation, with two values of cost and computation time. In DVRP results, we do not include the time calculation with an assumption that the calculation time to get the routes is in extent for real-time applications.

4.2 Simulation Results

Firstly, to verify the NLNM and the effectiveness of de-cranking in NLNM in this dissertation, nine customers with single vehicle was considered. Tables 4.2 and 4.3 show the results of NLNM and NLNM with de-cranking, respectively. The decreased rate of cost is shown by increasing the number of neighbor L in Figs. 4.3 and 4.4, respectively.

Next, we considered 100 customers random sample (R101) with a single vehicle. The simulation results have been given in Figs. 4.5 – 4.10 for three cases without/with de-cranking. The cost and computational time of simulation results are summarized in Table 4.4. Also, results for the multi vehicles have been presented in Figs 4.11 – 4.12 and Table 4.5.

Lastly, the de-cranking heuristic method and NLNM have applied to DVRP in considering five levels of degree of dynamic. The results are showed in Tables 4.9 – 4.14.

4.2.1 VRP

Table 4.2 Simulation results of VRP

| N=9 | Simulation cases | | | | | |
|-----|------------------|---------|--------|---------|--------|---------|
| | Case 1 | | Case 2 | | Case 3 | |
| L | cost | time(s) | cost | time(s) | cost | time(s) |
| 1 | 158 | 0.01 | 1086 | 0.01 | 1182 | 0.02 |
| 2 | 176 | 0.04 | 967 | 0.04 | 1351 | 0.03 |
| 3 | 158 | 0.14 | 1197 | 0.14 | 992 | 0.14 |
| 4 | 146 | 0.68 | 1104 | 0.68 | 1079 | 0.68 |
| 5 | 146 | 3.71 | 954 | 3.58 | 984 | 3.57 |
| 6 | 158 | 15.12 | 940 | 15.02 | 969 | 15.05 |
| 7 | 158 | 48.18 | 835 | 48.57 | 895 | 48.55 |
| 8 | 161 | 103.76 | 714 | 103.26 | 732 | 103.40 |
| 9 | 146 | 110.24 | 667 | 110.11 | 685 | 110.26 |

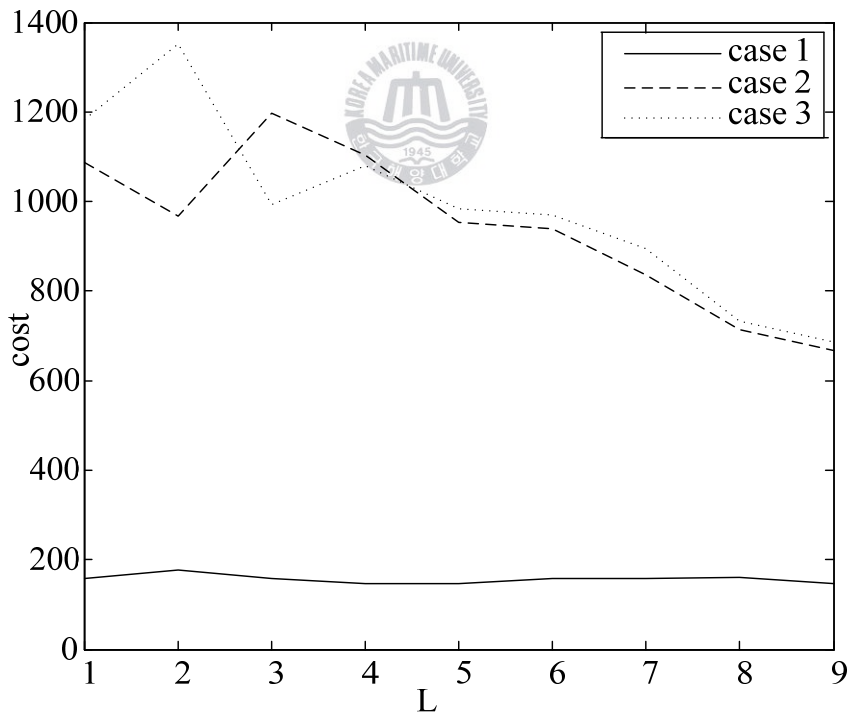


Fig. 4.3 Simulation results of NLNM

Table 4.3 Simulation results of VRP with de-cranking

| N=9 | Simulation cases | | | | | |
|-----|------------------|---------|--------|---------|--------|---------|
| | Case 1 | | Case 2 | | Case 3 | |
| L | cost | time(s) | cost | time(s) | cost | time(s) |
| 1 | 146 | 0.03 | 771 | 0.02 | 795 | 0.02 |
| 2 | 151 | 0.05 | 819 | 0.05 | 852 | 0.05 |
| 3 | 146 | 0.14 | 902 | 0.15 | 795 | 0.15 |
| 4 | 146 | 0.70 | 737 | 0.71 | 851 | 0.70 |
| 5 | 146 | 3.58 | 667 | 3.64 | 685 | 3.58 |
| 6 | 146 | 15.22 | 667 | 15.24 | 685 | 15.25 |
| 7 | 146 | 49.08 | 737 | 49.10 | 732 | 49.00 |
| 8 | 149 | 104.65 | 714 | 104.30 | 732 | 104.34 |
| 9 | 146 | 111.30 | 667 | 111.58 | 685 | 110.77 |

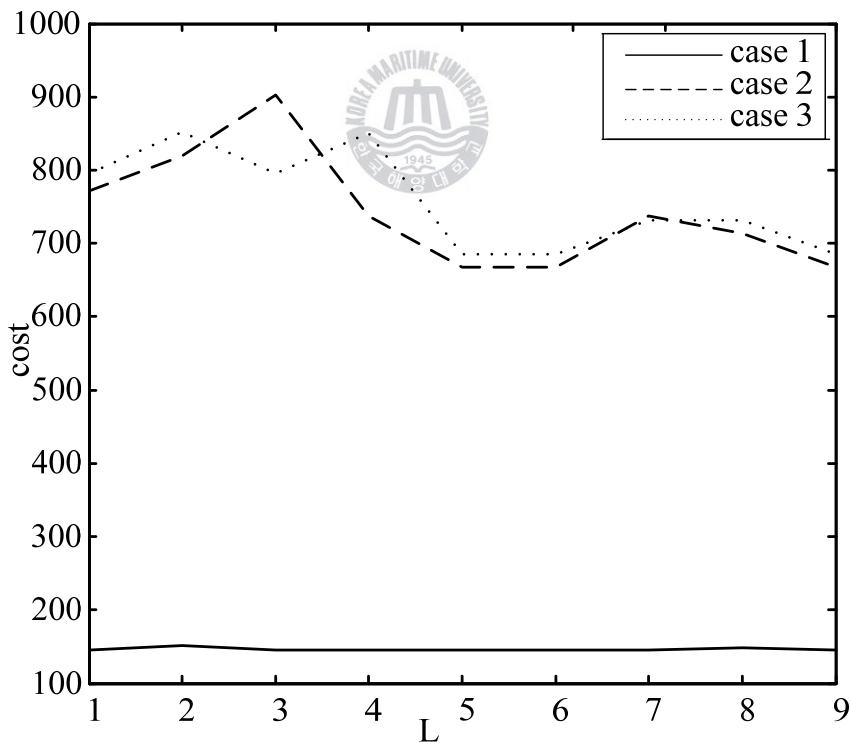


Fig. 4.4 Simulation results of NLNM with de-cranking

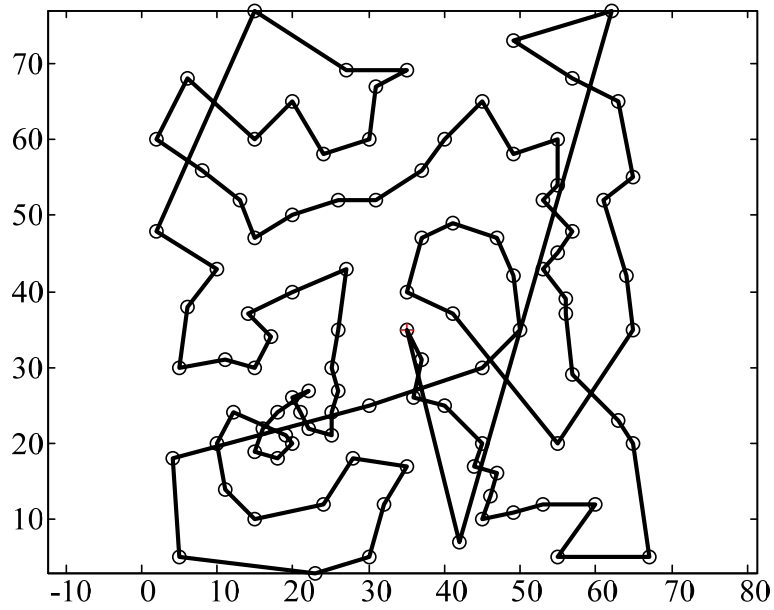


Fig. 4.5 VRP without de-cracking of sample R101 with $N=100$, $K=1$ and $L=1$, case 1

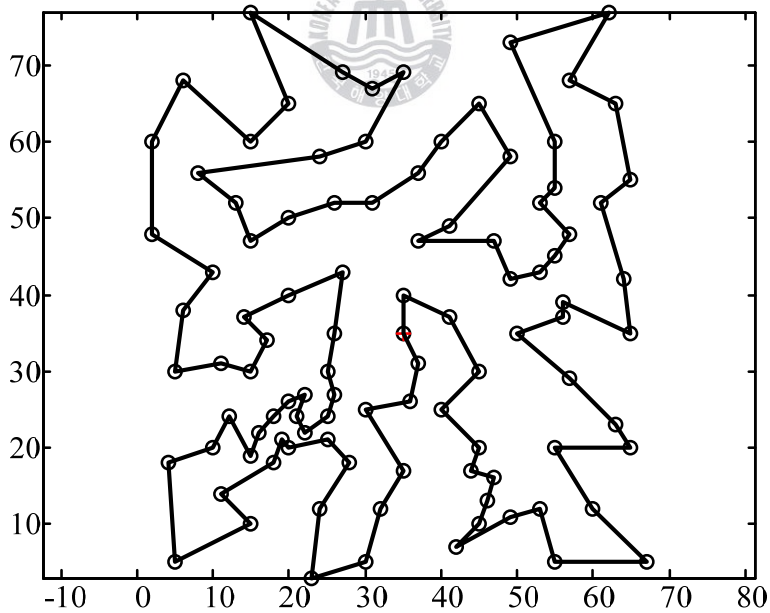


Fig. 4.6 VRP with de-cracking of sample R101 with $N=100$, $K=1$ and $L=1$, case 1

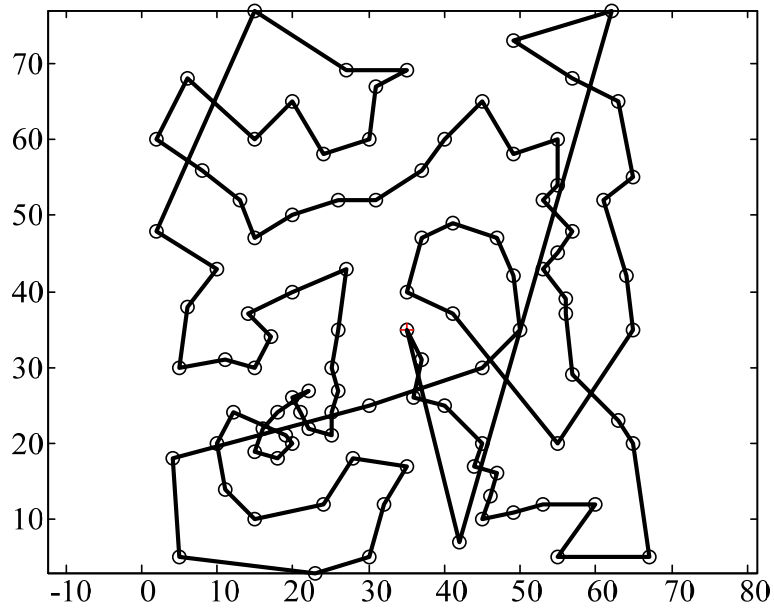


Fig. 4.7 VRP without de-cranking of sample R101 with $N=100$, $K=1$ and $L=1$, case 2

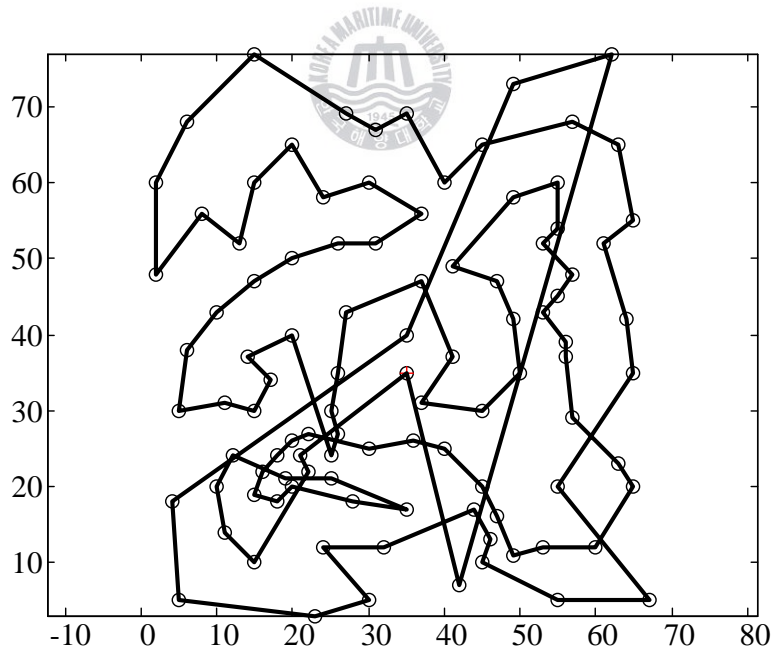


Fig. 4.8 VRP with de-cranking of sample R101 with $N=100$, $K=1$ and $L=1$, case 2

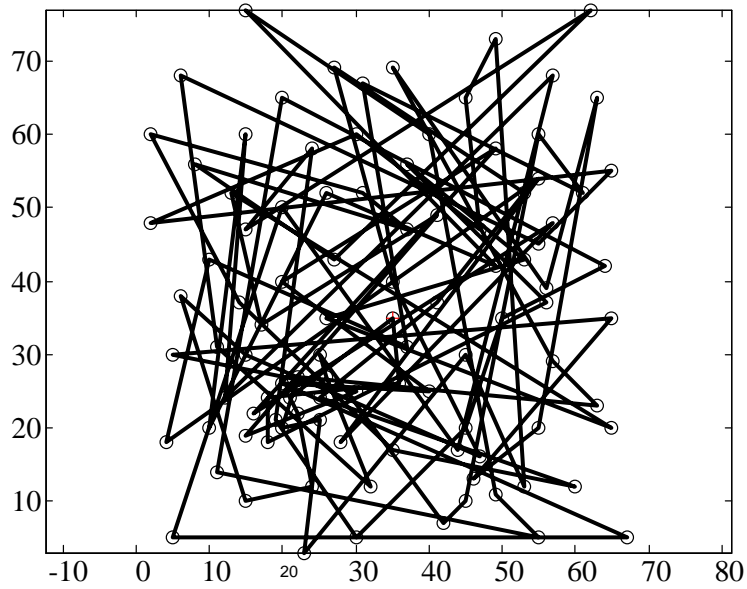


Fig. 4.9 VRP without de-cranking of sample R101 with $N=100$, $K=1$ and $L=1$, case 3

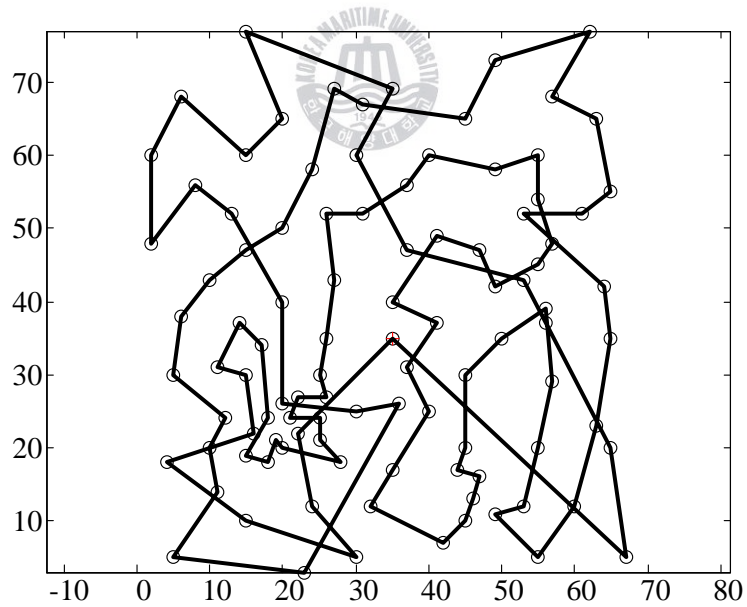


Fig. 4.10 VRP with de-cranking of sample R101 with $N=100$, $K=1$ and $L=1$, case 3

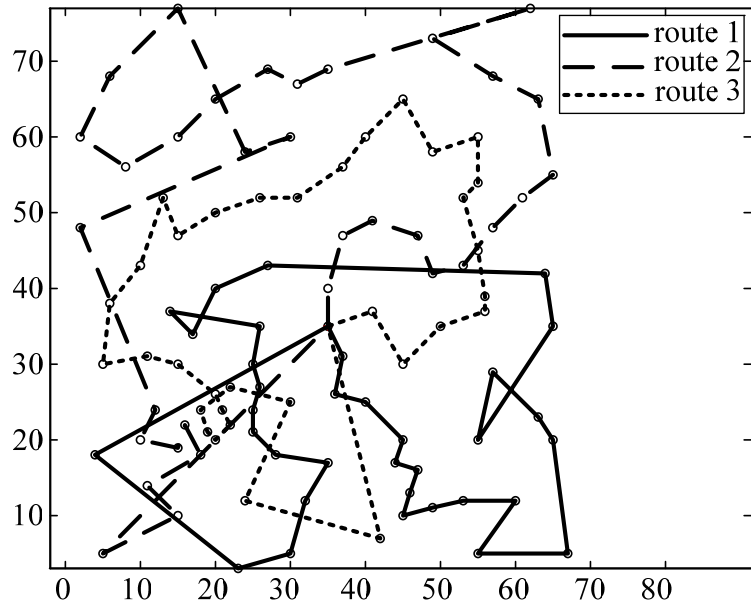


Fig. 4.11 VRP with sample R101 with $N=100$, $K=3$ and $L=1$, case 1

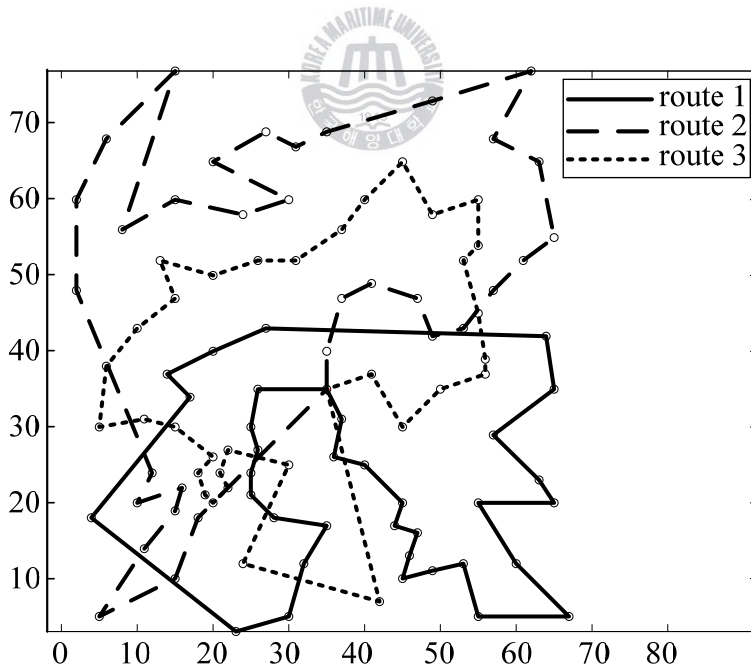


Fig. 4.12 VRP with de-crossing of sample R101 with $N=100$, $K=3$ and $L=1$, case 1

Table 4.4 Simulation results of VRP with single vehicle

| N=100 | Simulation cases | | | | | |
|------------------|------------------|---------|--------|---------|---------|---------|
| | Case 1 | | Case 2 | | Case 3 | |
| | cost | time(s) | cost | time(s) | cost | time(s) |
| without | 808 | 1.14 | 93,205 | 1.02 | 200,100 | 1.01 |
| with de-cranking | 678 | 33.03 | 76,699 | 28.50 | 81,091 | 37.91 |

Table 4.5 Simulation results of VRP with multi vehicles

| N=100 | | Simulation cases | | | | | |
|-------|------------------|------------------|---------|---------|---------|---------|---------|
| | | Case 1 | | Case 2 | | Case 3 | |
| k | | cost | time(s) | cost | time(s) | cost | time(s) |
| 1 | without | 808 | 1.14 | 93,205 | 1.02 | 200,100 | 1.01 |
| | with de-cranking | 678 | 33.03 | 76,699 | 28.50 | 81,091 | 37.91 |
| 2 | without | 624 | 1.09 | 134,360 | 0.98 | 204,250 | 0.98 |
| | with de-cranking | 555 | 57.14 | 116,980 | 98.88 | 120,350 | 98.69 |
| 3 | without | 556 | 1.19 | 177,090 | 1.00 | 241,010 | 0.99 |
| | with de-cranking | 509 | 164.91 | 155,430 | 143.80 | 161,530 | 250.48 |

Table 4.6 NLNM with de-cranking, case 1

| NLNM with total de-cranking | | | | |
|-----------------------------|---------|---------|---------|---------|
| Vehicles | Case 1 | | | |
| | cost | | time(s) | |
| | L=1 | L=2 | L=1 | L=2 |
| 1 | 678.035 | 685.472 | 8.9492 | 24.0634 |
| 2 | 550.599 | 625.937 | 2.4602 | 19.7876 |
| 3 | 509.361 | 482.076 | 2.3239 | 20.4968 |
| 4 | 495.389 | 491.368 | 1.418 | 19.7459 |
| 5 | 422.428 | 392.935 | 1.2105 | 17.7639 |
| 6 | 439.267 | 341.659 | 2.6854 | 18.312 |
| 7 | 418.574 | 432.645 | 1.2025 | 17.7689 |
| 8 | 419.134 | 460.214 | 1.7907 | 17.9893 |
| 9 | 364.9 | 465.565 | 0.9603 | 20.3855 |
| 10 | 364.642 | 367.92 | 0.9468 | 17.4402 |
| 11 | 414.098 | 377.779 | 1.0881 | 22.2366 |
| 12 | 378.182 | 460.095 | 1.3801 | 18.7346 |
| 13 | 332.175 | 370.38 | 4.1263 | 31.4744 |
| 14 | 423.602 | 362.844 | 1.1566 | 27.6806 |
| 15 | 375.03 | 397.568 | 5.158 | 20.5543 |
| 16 | 370.941 | 447.828 | 1.7467 | 18.0479 |
| 17 | 305.709 | 286.122 | 12.5557 | 70.1792 |
| 18 | 334.451 | 313.86 | 6.037 | 58.3791 |
| 19 | 360.977 | 349.976 | 1.8087 | 48.2011 |
| 20 | 289.361 | 364.193 | 0.6033 | 40.2694 |

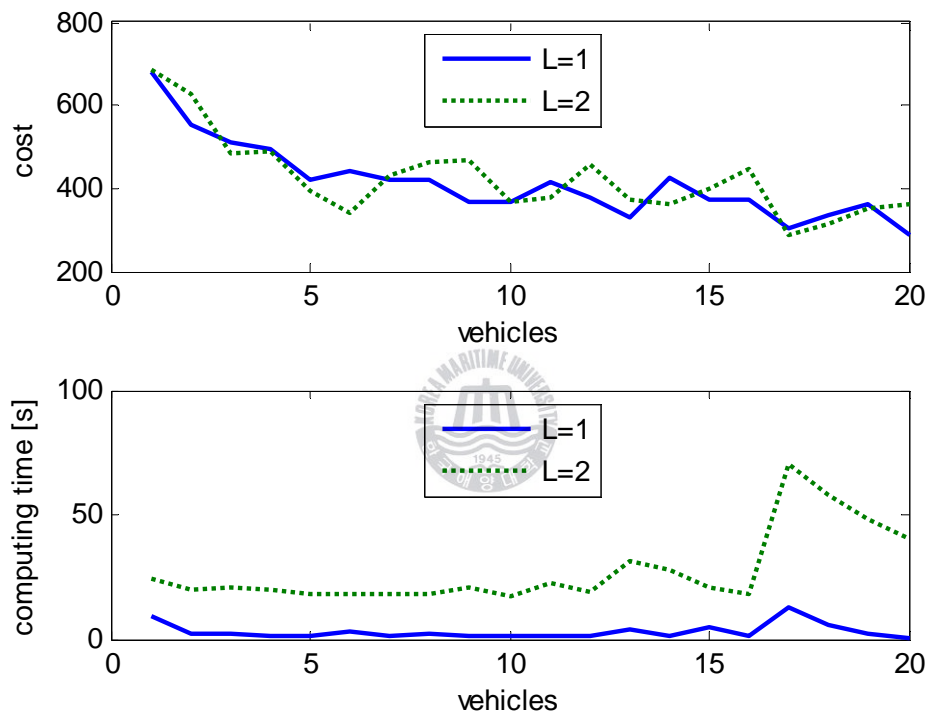


Fig. 4.13 NLNM with de-cranking, case 1

Table 4.7 NLNM with de-cranking, case 2

| NLNM with total de-cranking | | | | |
|-----------------------------|------------------------|--------|---------------|---------|
| Vehicles | Case 2 | | | |
| | cost [$\times 10^4$] | | time(s) | |
| | L=1 | L=2 | L=1 | L=2 |
| 1 | 7.6699 | 7.9564 | 9.5428 | 27.2364 |
| 2 | 3.6582 | 3.4235 | 3.8253 | 21.2531 |
| 3 | 2.1846 | 2.3721 | 2.9591 | 22.1382 |
| 4 | 1.4352 | 1.9221 | 2.7764 | 19.5542 |
| 5 | 1.0301 | 1.033 | 1.7481 | 18.2923 |
| 6 | 1.1541 | 1.0954 | 2.3151 | 19.6413 |
| 7 | 0.7708 | 0.8149 | 2.2661 | 19.1046 |
| 8 | 0.849 | 0.798 | 2.1326 | 20.2376 |
| 9 | 0.4248 | 1.2931 | 1.7051 | 22.0297 |
| 10 | 0.2357 | 0.2643 | 1.2057 | 17.9264 |
| 11 | 0.2969 | 1.8085 | 1.3721 | 24.5302 |
| 12 | 0.5569 | 0.5568 | 2.0351 | 18.6155 |
| 13 | 1.1865 | 4.4533 | 4.8966 | 40.8315 |
| 14 | 0.3395 | 2.7841 | 1.532 | 33.5581 |
| 15 | 1.3995 | 1.4938 | 7.5675 | 22.898 |
| 16 | 0.566 | 0.4161 | 2.4024 | 18.6891 |
| 17 | 2.4105 | 9.5105 | 15.4802 | 103.663 |
| 18 | 1.4896 | 7.7354 | 6.0711 | 85.3255 |
| 19 | 0.524 | 6.2626 | 2.2725 | 58.2623 |
| 20 | 0.0668 | 4.575 | 0.6736 | 52.8186 |

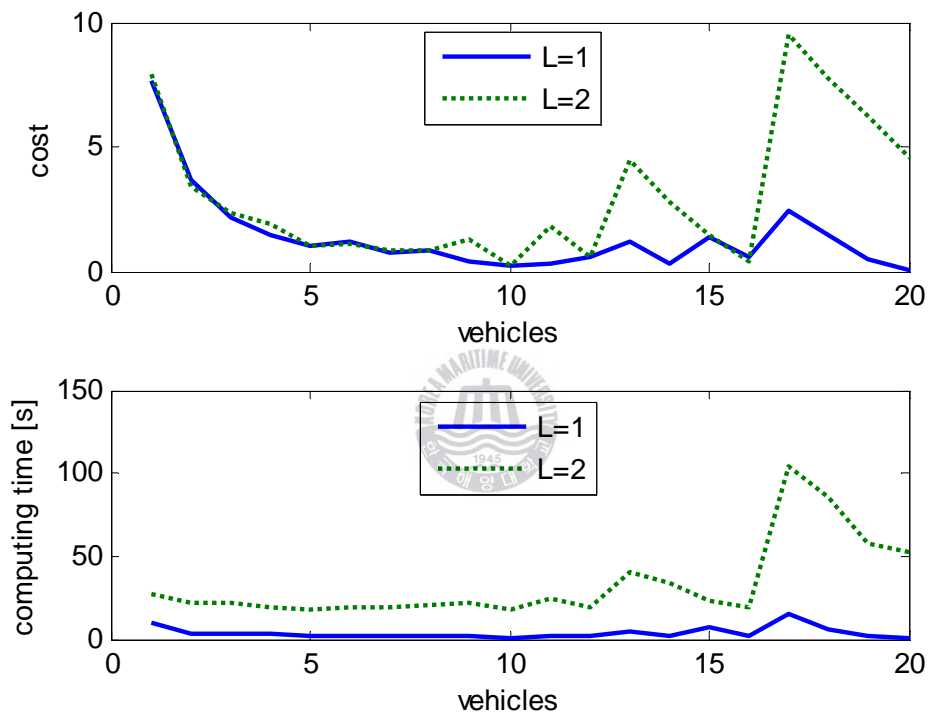


Fig. 4.14 NLNM with de-cranking, case 2

Table 4.8 NLNM with de-cranking, case 3

| NLNM with total de-cranking | | | | |
|-----------------------------|------------------------|--------|---------|---------|
| Vehicles | Case 3 | | | |
| | cost [$\times 10^4$] | | time(s) | |
| | L=1 | L=2 | L=1 | L=2 |
| 1 | 8.1091 | 7.7574 | 11.6959 | 32.3558 |
| 2 | 3.5885 | 3.5879 | 6.5484 | 23.138 |
| 3 | 2.2216 | 2.3999 | 3.9412 | 20.4269 |
| 4 | 1.5125 | 1.6833 | 2.4247 | 19.8097 |
| 5 | 1.1419 | 0.9988 | 2.011 | 18.1762 |
| 6 | 1.1244 | 0.9313 | 2.3239 | 18.9247 |
| 7 | 0.7778 | 0.5902 | 2.0043 | 18.3404 |
| 8 | 0.7636 | 0.6106 | 1.7986 | 18.3663 |
| 9 | 0.3942 | 1.0601 | 1.4119 | 21.4881 |
| 10 | 0.2982 | 0.2853 | 0.6826 | 17.9081 |
| 11 | 0.3145 | 1.166 | 0.8877 | 26.1126 |
| 12 | 0.377 | 0.3779 | 1.4603 | 17.6651 |
| 13 | 0.866 | 3.4091 | 4.0198 | 37.5516 |
| 14 | 0.3001 | 2.0996 | 0.9557 | 25.3536 |
| 15 | 0.9593 | 1.0464 | 6.3958 | 19.3937 |
| 16 | 0.3956 | 0.3744 | 1.4025 | 17.6562 |
| 17 | 2.1249 | 8.5678 | 6.7622 | 69.3687 |
| 18 | 1.159 | 6.941 | 7.4689 | 44.1398 |
| 19 | 0.5272 | 5.5331 | 1.9583 | 37.2734 |
| 20 | 0.2736 | 4.1551 | 0.4799 | 31.8757 |

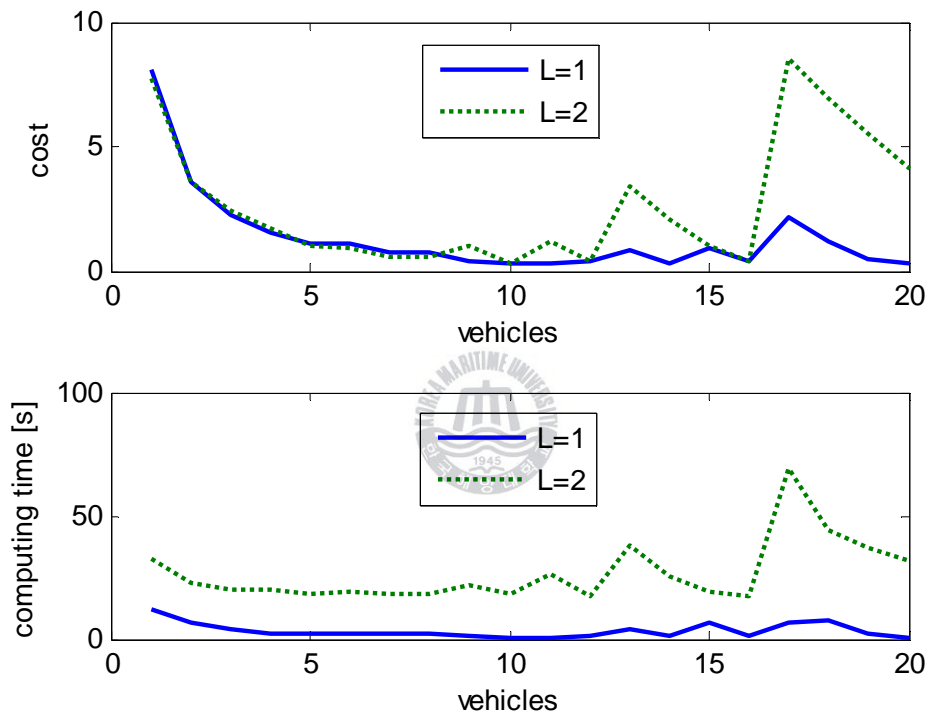


Fig. 4.15 NLNM with de-cranking, case 3

4.2.2 DVRP

Table 4.9 NLNM (L=1) with de-cranking for DVRP, case 1

| NLNM with L=1 and total de-cranking, case 1 | | | | | |
|---|--------------------------|----------|----------|----------|----------|
| Vehicle No. | Degree of dynamics (DoD) | | | | |
| | 1[$\times 10^3$] | 2 | 3 | 4 | 5 |
| 1 | 0.7999* | 792.547 | 744.891 | 704.476 | 694.252 |
| 2 | 0.8673 | 638.518* | 634.12 | 674.929 | 557.479 |
| 3 | 1.0505 | 707.356 | 601.738 | 664.42 | 583.989 |
| 4 | 1.1606 | 669.149 | 526.406 | 520.805 | 530.316 |
| 5 | 1.1221 | 663.453 | 423.410* | 492.380* | 453.745* |
| 6 | 0.9315 | 760.063 | 504.241 | 501.639 | 492.354 |
| 7 | 1.115 | 787.466 | 571.434 | 463.121 | 489.873 |
| 8 | 0.9158 | 719.746 | 580.241 | 525.235 | 440.616 |
| 9 | 1.1599 | 771.062 | 600.365 | 488.605 | 426.865 |
| 10 | 1.1387 | 692.937 | 621.858 | 594.162 | 481.424 |
| 11 | 0.8145 | 663.569 | 548.229 | 552.947 | 449.609 |
| 12 | 0.891 | 690.205 | 546.242 | 474.807 | 481.232 |
| 13 | 1.1512 | 686.542 | 521.893 | 528.724 | 580.306 |
| 14 | 0.8883 | 583.587 | 559.923 | 526.175 | 469.59 |
| 15 | 0.8781 | 674.48 | 611.909 | 614.785 | 511.787 |
| 16 | 0.9475 | 632.034 | 528.398 | 487.031 | 462.748 |
| 17 | 1.1341 | 622.312 | 558.825 | 488.493 | 475.631 |
| 18 | 0.8566 | 667.652 | 477.512 | 539.408 | 468.119 |
| 19 | 0.7357 | 620.602 | 546.507 | 506.93 | 426.456 |
| 20 | 0.8707 | 610.13 | 470.786 | 476.716 | 447.74 |

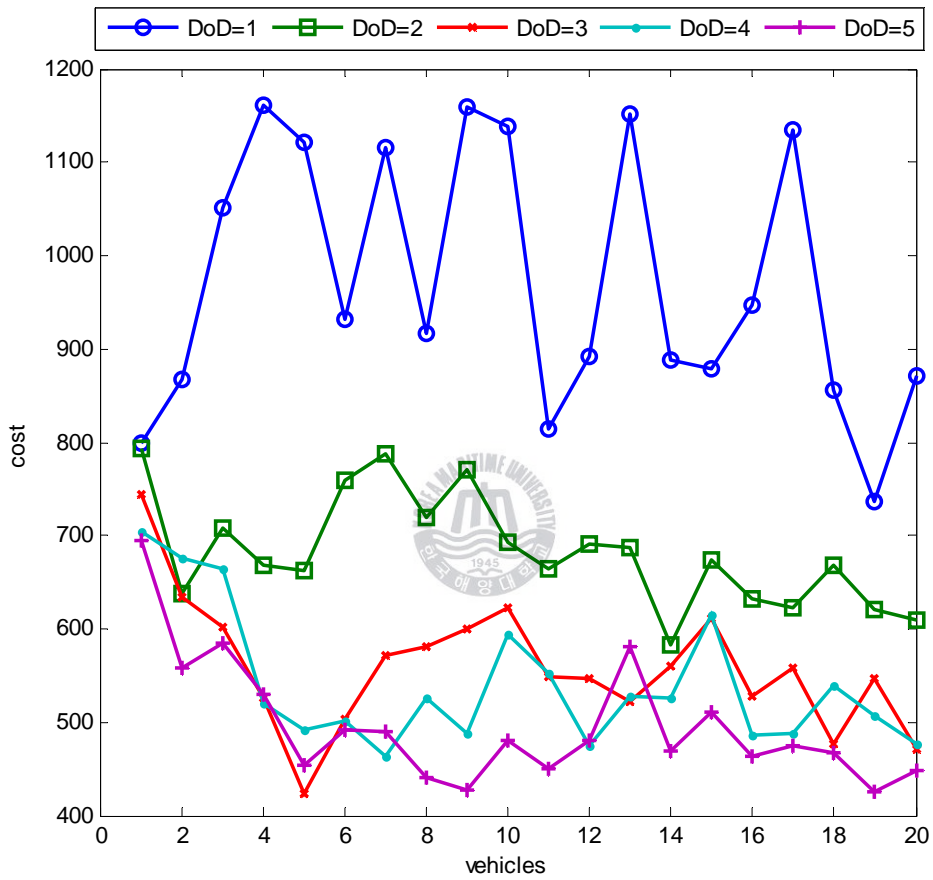


Fig. 4.16 NLNM (L=1) with de-cranking for DVVRP, case 1

Table 4.10 NLNM (L=1) with de-cranking for DVRP, Case 2

| NLNM with L=1 and total decracking, case 2 | | | | | |
|--|--------------------------|--------------------|--------------------|--------------------|--------------------|
| Vehicle No. | Degree of dynamics (DoD) | | | | |
| | 1[$\times 10^3$] | 2[$\times 10^3$] | 3[$\times 10^3$] | 4[$\times 10^3$] | 5[$\times 10^3$] |
| 1 | 1.7909 | 1.4497 | 1.1995 | 1.0503 | 1.007 |
| 2 | 0.8031 | 0.6183 | 0.6019 | 0.5045 | 0.5228 |
| 3 | 0.8887 | 0.4198 | 0.3793 | 0.3638 | 0.3325 |
| 4 | 0.8705 | 0.3259 | 0.2627 | 0.2303 | 0.2377 |
| 5 | 0.8954 | 0.3563 | 0.1677 | 0.1852 | 0.1931 |
| 6 | 1.146 | 0.3689 | 0.2275 | 0.1769 | 0.1523 |
| 7 | 1.0794 | 0.379 | 0.2005 | 0.1086 | 0.1196 |
| 8 | 1.3416 | 0.48 | 0.1696 | 0.1607 | 0.1016 |
| 9 | 1.3481 | 0.5204 | 0.3213 | 0.2501 | 0.0773 |
| 10 | 1.578 | 0.6314 | 0.3468 | 0.2582 | 0.0791 |
| 11 | 1.7025 | 0.8221 | 0.414 | 0.3 | 0.1304 |
| 12 | 1.8769 | 0.9227 | 0.2597 | 0.3228 | 0.1263 |
| 13 | 1.9619 | 1.0167 | 0.4836 | 0.5176 | 0.1525 |
| 14 | 2.1785 | 1.2298 | 0.7437 | 0.349 | 0.3477 |
| 15 | 2.2812 | 1.1563 | 0.4885 | 0.6805 | 0.2174 |
| 16 | 2.5611 | 1.2931 | 0.5264 | 0.4036 | 0.1803 |
| 17 | 2.4697 | 1.2007 | 0.9426 | 0.3113 | 0.1871 |
| 18 | 2.7974 | 1.8623 | 0.9644 | 0.878 | 0.1823 |
| 19 | 3.0317 | 1.7084 | 0.5765 | 0.8247 | 0.6834 |
| 20 | 3.1948 | 2.0381 | 0.6897 | 0.5802 | 0.5614 |

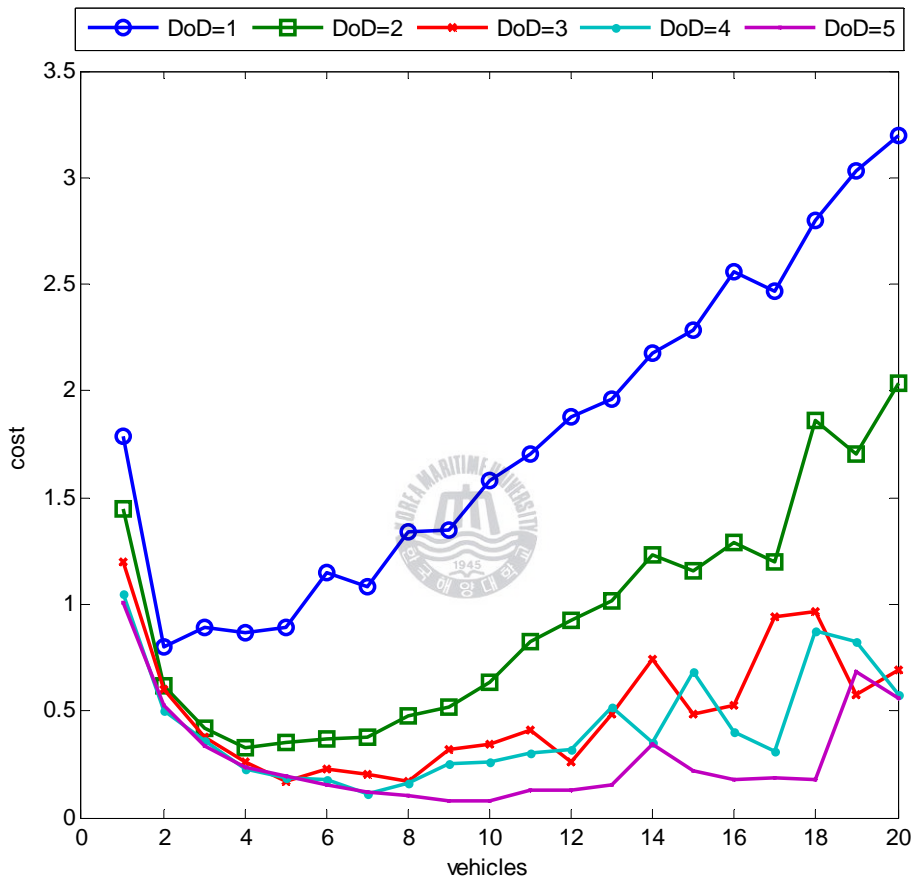


Fig. 4.17 NLNM (L=1) with de-cranking for DVRP, case 2

Table 4.11 NLNM (L=1) with de-cranking for DVRP, case 3

| NLNM with L=1 and total de-cranking, case 3 | | | | | |
|---|--------------------------|---------------------|---------------------|---------------------|---------------------|
| Vehicle No. | Degree of dynamics (DoD) | | | | |
| | 1 [$\times 10^3$] | 2 [$\times 10^3$] | 3 [$\times 10^3$] | 4 [$\times 10^3$] | 5 [$\times 10^3$] |
| 1 | 1.7074 | 1.29 | 1.1622 | 1.028 | 1.0941 |
| 2 | 0.884 | 0.719 | 0.6644 | 0.5596 | 0.5569 |
| 3 | 0.939 | 0.4559 | 0.4444 | 0.377 | 0.3803 |
| 4 | 0.9689 | 0.3416 | 0.2791 | 0.2643 | 0.269 |
| 5 | 0.9766 | 0.3656 | 0.1948 | 0.1891 | 0.1923 |
| 6 | 1.2494 | 0.3954 | 0.1739 | 0.1957 | 0.1807 |
| 7 | 1.124 | 0.3806 | 0.2428 | 0.1384 | 0.1244 |
| 8 | 1.6161 | 0.366 | 0.1889 | 0.1947 | 0.1171 |
| 9 | 1.2995 | 0.4127 | 0.2638 | 0.2112 | 0.0746 |
| 10 | 1.6907 | 0.4772 | 0.2992 | 0.1561 | 0.0991 |
| 11 | 2.2704 | 0.6867 | 0.2512 | 0.1594 | 0.1139 |
| 12 | 2.0806 | 0.8098 | 0.3525 | 0.1774 | 0.1189 |
| 13 | 1.8676 | 0.7933 | 0.5976 | 0.2541 | 0.1683 |
| 14 | 2.9503 | 0.5548 | 0.495 | 0.4124 | 0.1049 |
| 15 | 2.7797 | 1.1246 | 0.7713 | 0.2305 | 0.1813 |
| 16 | 2.6446 | 0.9826 | 0.7405 | 0.6618 | 0.1411 |
| 17 | 2.5027 | 0.8393 | 0.6744 | 0.5145 | 0.2757 |
| 18 | 4.3154 | 1.529 | 0.4034 | 0.3554 | 0.2701 |
| 19 | 4.2971 | 1.3349 | 1.2315 | 0.2632 | 0.1678 |
| 20 | 4.2139 | 1.3078 | 1.0794 | 1.0211 | 0.6979 |

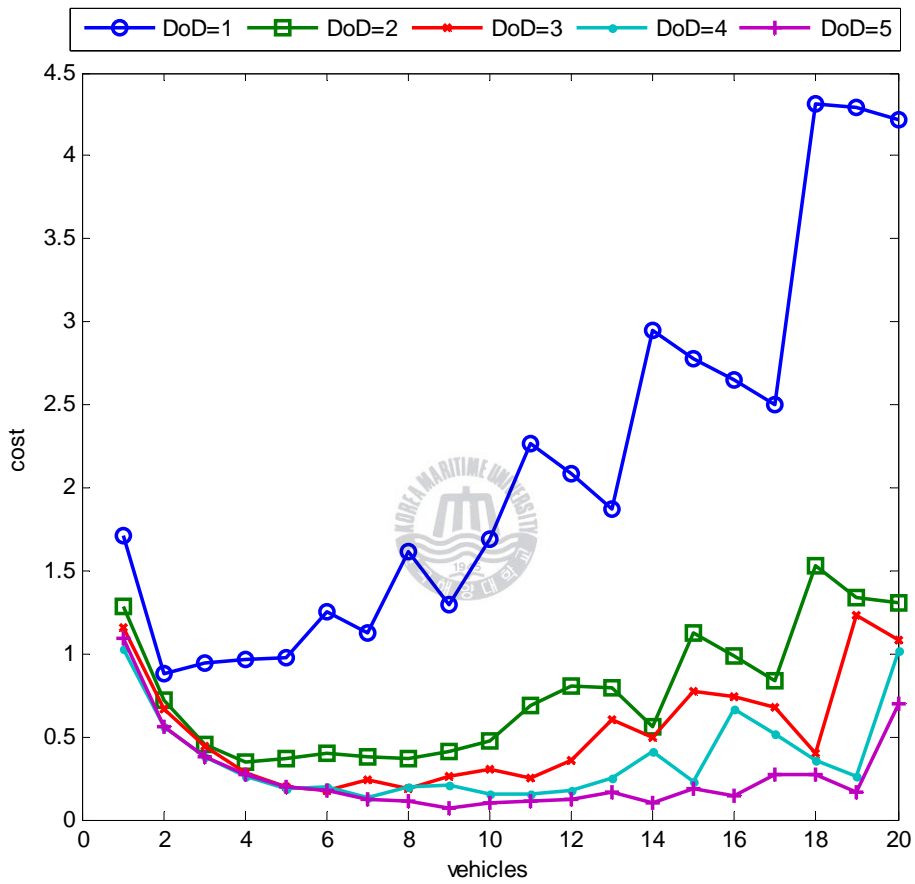


Fig. 4.18 NLNM (L=1) with de-cranking for DVRP, case 3

Table 4.12 NLNM (L=2) with de-cranking for DVRP, case 1

| NLNM with L=2 and total de-cranking, case 1 | | | | | |
|---|--------------------------|----------|----------|----------|----------|
| Vehicle No. | Degree of dynamics (DoD) | | | | |
| | 1[×10 ³] | 2 | 3 | 4 | 5 |
| 1 | 0.8549 | 783.412 | 738.746 | 721.272 | 745.886 |
| 2 | 0.7897* | 667.019 | 665.731 | 635.265 | 659.701 |
| 3 | 0.9281 | 648.876 | 658.791 | 607.575 | 591.6 |
| 4 | 0.9036 | 574.693* | 557.599 | 546.03 | 519.611 |
| 5 | 0.8586 | 629.435 | 539.809* | 517.472* | 489.482* |
| 6 | 0.8059 | 710.936 | 566.959 | 513.634 | 519.553 |
| 7 | 0.855 | 626.407 | 524.082 | 508.452 | 443.273 |
| 8 | 0.8071 | 587.392 | 531.586 | 467.939 | 432.383 |
| 9 | 0.7244 | 598.242 | 543.549 | 501.253 | 436.463 |
| 10 | 0.8365 | 583.762 | 482.4 | 467.868 | 414.792 |
| 11 | 0.718 | 646.608 | 533.979 | 480.3 | 408.362 |
| 12 | 0.7344 | 528.683 | 511.414 | 430.208 | 465.858 |
| 13 | 0.6761 | 563.974 | 410.996 | 408.763 | 432.339 |
| 14 | 0.7763 | 528.133 | 493.224 | 470.221 | 396.624 |
| 15 | 0.7874 | 540.789 | 442.006 | 486.158 | 358.545 |
| 16 | 0.7571 | 588.636 | 442.869 | 444.184 | 392.029 |
| 17 | 0.6434 | 539.21 | 478.764 | 399.817 | 399.338 |
| 18 | 0.6712 | 552.888 | 427.028 | 359.067 | 369.756 |
| 19 | 0.7251 | 504.627 | 473.313 | 432.514 | 380.835 |
| 20 | 0.6855 | 509.49 | 432.306 | 394.342 | 356.022 |

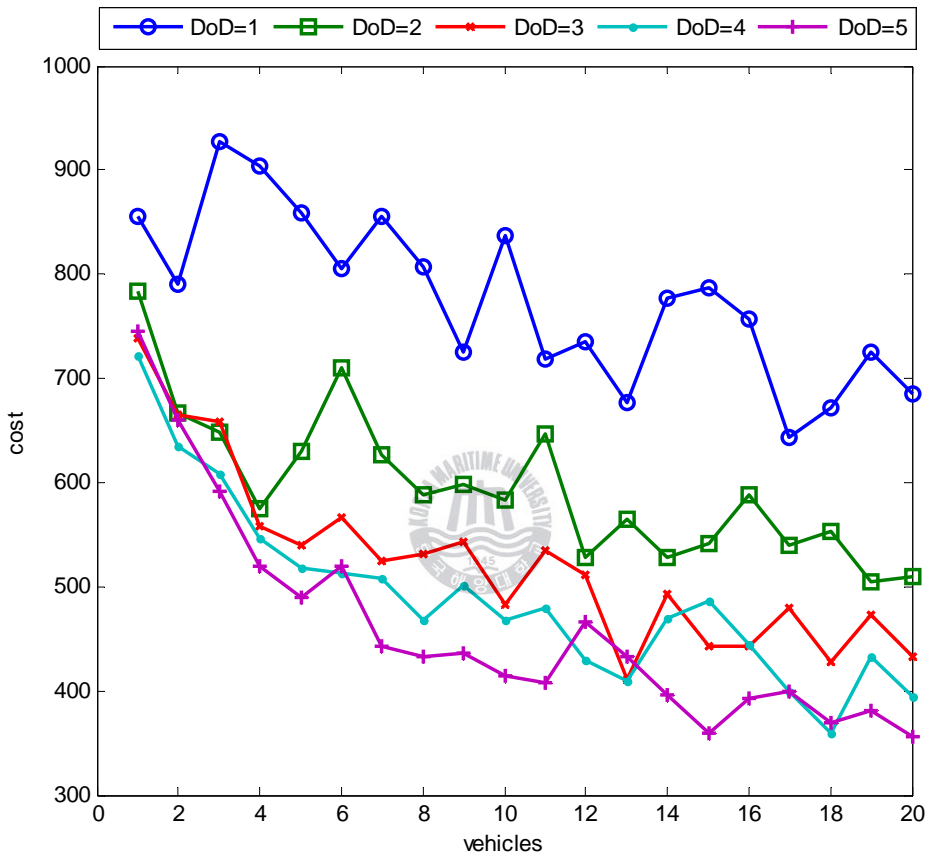


Fig. 4.19 NLNM (L=2) with de-cranking for DVRP, case 1

Table 4.13 NLNM (L=2) with de-cranking for DVRP, case 2

| NLNM with L=2 and total de-cranking, case 2 | | | | | |
|---|--------------------------|---------------------|---------------------|---------------------|---------------------|
| Vehicle No. | Degree of dynamics (DoD) | | | | |
| | 1 [$\times 10^3$] | 2 [$\times 10^3$] | 3 [$\times 10^3$] | 4 [$\times 10^3$] | 5 [$\times 10^3$] |
| 1 | 1.6738 | 1.2264 | 1.0835 | 1.0524 | 0.9899 |
| 2 | 0.8886 | 0.7118 | 0.6226 | 0.5475 | 0.5003 |
| 3 | 0.9418 | 0.4339 | 0.3534 | 0.3596 | 0.3403 |
| 4 | 0.9839 | 0.3717 | 0.2764 | 0.2315 | 0.2258 |
| 5 | 1.11 | 0.415 | 0.2204 | 0.2494 | 0.2373 |
| 6 | 1.2034 | 0.4516 | 0.3302 | 0.2377 | 0.235 |
| 7 | 1.5962 | 0.6572 | 0.3068 | 0.0916 | 0.1498 |
| 8 | 1.6974 | 0.8038 | 0.3722 | 0.2584 | 0.2314 |
| 9 | 2.0214 | 0.8475 | 0.5403 | 0.1161 | 0.2633 |
| 10 | 1.9603 | 0.9618 | 0.6131 | 0.4088 | 0.3725 |
| 11 | 2.3633 | 1.2967 | 0.8437 | 0.6812 | 0.5096 |
| 12 | 2.4354 | 1.1366 | 0.917 | 0.6881 | 0.6762 |
| 13 | 3.1755 | 1.7489 | 0.9006 | 0.9015 | 0.694 |
| 14 | 3.317 | 1.2494 | 0.2888 | 0.7548 | 0.7296 |
| 15 | 3.7307 | 1.4485 | 1.1387 | 1.3199 | 0.8862 |
| 16 | 3.5939 | 2.623 | 1.1302 | 1.0904 | 1.0711 |
| 17 | 4.5277 | 2.6717 | 0.3529 | 0.9707 | 0.7584 |
| 18 | 4.6362 | 1.8478 | 1.6203 | 1.6591 | 0.9849 |
| 19 | 5.0597 | 2.1007 | 1.6371 | 1.4789 | 1.4479 |
| 20 | 5.099 | 2.2736 | 1.8613 | 1.559 | 1.4296 |

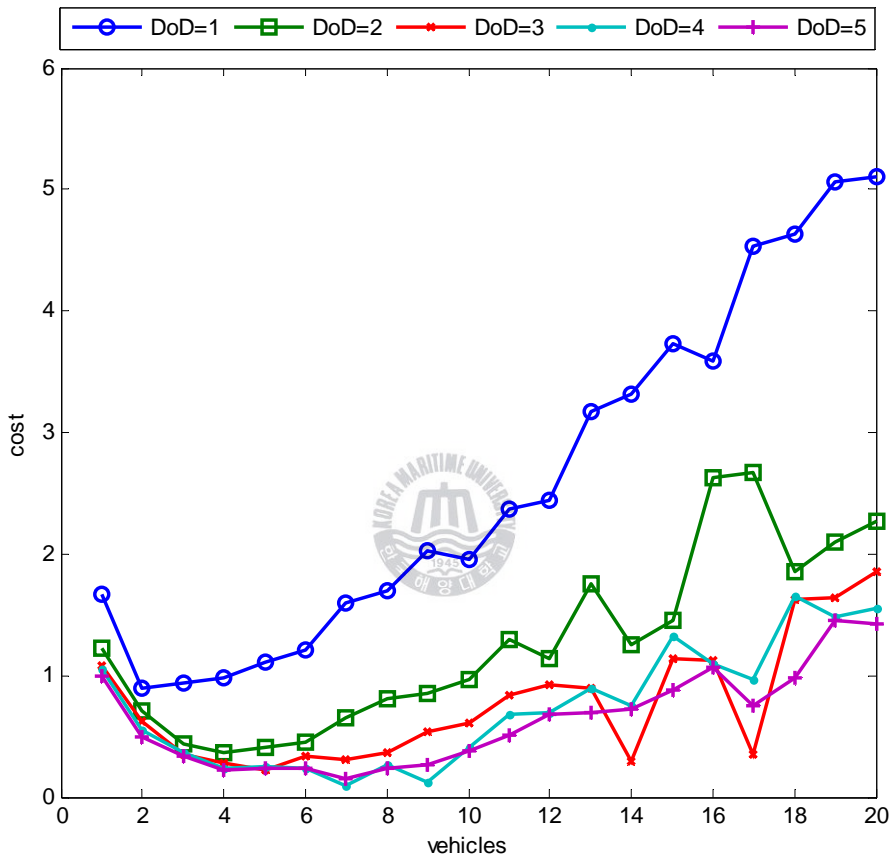


Fig. 4.20 NLNM (L=2) with de-cranking for DVRP, case 2

Table 4.14 NLNM (L=2) with de-cranking for DVRP, case 3

| NLNM with L=2 and total de-cranking, case 3 | | | | | |
|---|--------------------------|---------------------|---------------------|---------------------|---------------------|
| Vehicle No. | Degree of dynamics (DoD) | | | | |
| | 1 [$\times 10^3$] | 2 [$\times 10^3$] | 3 [$\times 10^3$] | 4 [$\times 10^3$] | 5 [$\times 10^3$] |
| 1 | 1.7316 | 1.3173 | 1.1333 | 1.1391 | 0.9933 |
| 2 | 0.9079 | 0.6718 | 0.5771 | 0.5665 | 0.5128 |
| 3 | 1.0453 | 0.4889 | 0.4303 | 0.384 | 0.3384 |
| 4 | 1.1676 | 0.4087 | 0.2741 | 0.2435 | 0.251 |
| 5 | 1.4192 | 0.4278 | 0.2698 | 0.234 | 0.2453 |
| 6 | 1.5078 | 0.5777 | 0.3124 | 0.2658 | 0.251 |
| 7 | 1.8232 | 0.5607 | 0.3664 | 0.3013 | 0.2039 |
| 8 | 1.9352 | 0.7714 | 0.1918 | 0.3555 | 0.2405 |
| 9 | 2.583 | 0.6944 | 0.4901 | 0.3167 | 0.256 |
| 10 | 2.4263 | 0.8207 | 0.5271 | 0.4937 | 0.2879 |
| 11 | 3.1427 | 1.2431 | 0.6784 | 0.6914 | 0.3197 |
| 12 | 3.1269 | 1.0495 | 0.9232 | 0.8118 | 0.4252 |
| 13 | 4.2559 | 1.5439 | 1.2137 | 0.6278 | 0.6432 |
| 14 | 4.3813 | 1.2052 | 0.9479 | 0.9977 | 0.5291 |
| 15 | 4.462 | 1.9724 | 0.9932 | 0.8249 | 0.9859 |
| 16 | 4.3829 | 2.0976 | 0.3342 | 0.9359 | 1.1114 |
| 17 | 6.7284 | 2.0286 | 1.9866 | 1.8218 | 0.9411 |
| 18 | 6.9354 | 1.802 | 1.7313 | 1.6741 | 1.6208 |
| 19 | 7.0952 | 3.2489 | 1.5991 | 1.5752 | 1.3694 |
| 20 | 7.3089 | 3.3211 | 0.383 | 1.5461 | 1.4135 |

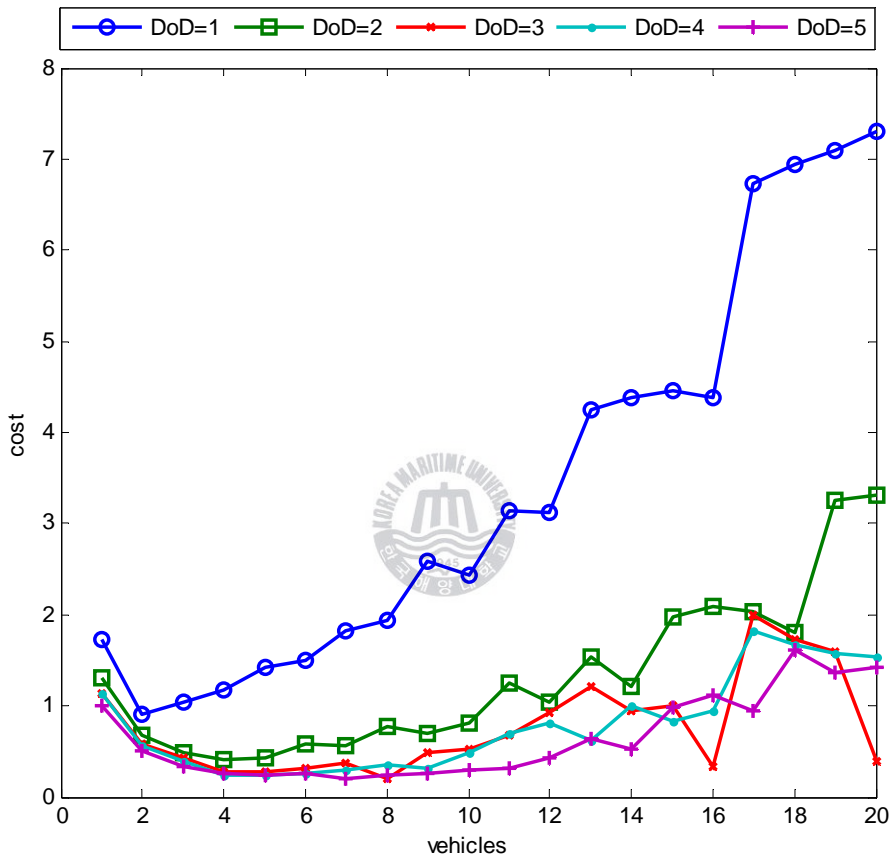


Fig. 4.21 NLNM (L=2) with de-cranking for DVRP, case 3

4.3 Comments and Remarks

As shown in Table 4.2 and Fig. 4.3, the costs decrease when increasing the number L of neighbors. The exact solution is not guaranteed to be obtained by NLNM. However, NLNM with de-cranking can quickly find a near-exact solution as shown in Table 4.3 and Fig. 4.4.

Figs. 4.5, 4.7 and 4.9 show the results of NLNM without de-cranking for 100 customers. Figs. 4.6, 4.8 and 4.10 show the results of NLNM with de-cranking.

Fig. 4.5 shows a completely free cranking in the case in which the vehicle visits all customers and returns to depot. In Figs. 4.8 and 4.10, crankies appear in the routes because the re-arrangement depended on the objective function in consideration of penalties. So, the de-cranking can reduce the cost space, not in ordinary Euclidean space.

In Table 4.4, the cost of NLNM with de-cranking is reduced more than that of NLNM. So, the NLNM with de-cranking is a more powerful means to get a better solution than NLNM. However, the comparison results with exact solution could not be shown in this dissertation because the computer took a long time in the calculation of simulation.

The simulation of multiple vehicles is applied to 100 customers. To show the effectiveness of NLNM with de-cranking in the multi-vehicle situation (case 1), $L = 1$ is considered and the results are shown in Figs 27 and 28. Moreover, the cost and computational time of simulation results with multiple vehicles are summarized in Table 4.5 for three cases with changing number of vehicles.

As a result, the cost of the objective function can be reduced by using the NLNM with de-cranking more than NLNM, but the computation of NLNM with de-cranking takes a longer time than that of NLNM. It is certain because time operation of two heuristic procedures takes more time than just one. But if we compare the computation time of NLNM with $L = 1, 2$ plus de-cranking operation time with the highest time in the cases of just nine locations as in Tables 4.2 and 4.3, they has shorter computation time. These results are important and remarkable to think about applying the de-cranking heuristic method and NLNM with short length of L .

More results with multi vehicle shown in here with total de-crossing to the routes in Tables 4.6 – 4.8. It shows that the longer L might not effective in the cases. With $L = 1$, we can get acceptable solution in a very short time. For example, for VRP, in case 1 (Table 4.1), cost = 289.361 with computing time of 0.6 second in somewhat better the cost = 286.122 with computing time of 70.2 second. Case 2 and case 3 show the absolute advantageousness of NLNN with $L = 1$ combined with de-cranking procedure.

These methods also has effectiveness in dynamic context, and suitable for high degree of dynamics. We can see the results highlighted from Tables 4.9 – 4.14 that the good solutions can get with number of vehicles less than 10 units with both $L = 1$ and $L = 2$ and in all degree of dynamics. Except Table 4.9 and 4.12 to the case 1, with $L = 1$ and $L = 2$, which is just touring around the locations without any costing but traveling cost, the require of vehicles is greater than 12 units to get the good solutions, but these are minor cases and near the

good solutions (indicated with star * beside the value) can be gotten with no greater than 5 units.



Chapter 5 Conclusions

In this dissertation, the de-cranking heuristic methods and NLNM have been proposed to find a suboptimal routing solution for the vehicle routing problem involving multiple vehicles. The de-cranking procedure applied to solution routes from NLNM gives a better solution and takes a shorter computation time than solutions involving long string neighbors from NLNM.

The proposed methods have been useful for adaptation to real VPR with real-time scheduling. Also, the methods have been suitable for the DVRP with multi-vehicles and high degree of dynamics.

In future, applying these methods to other combinatorial optimization problems should do more tests.

Published Papers:

- [1] Kim, H.S. and Tran-Ngoc, H.S.(2009), “Nearest L - Neighbor Method with De-crossing in Vehicle Routing Problem”, International Journal of Navigation and Port Research, Vol. 33, No. 2, pp. 143~151.
- [2] Kim, H.S. and Tran-Ngoc, H.S.(2009), “De-Cranking the Routes to Solve the Dynamic Vehicle Routing Problem”, Technical Report, Department of Logistics, Korea Maritime University.
- [3] Kim, H.S. and Tran-Ngoc, H.S.(2009), “Modeling Vehicle Routing Problem with Pair Pickup-Delivery Operations “, Conference of KINPR, pp. 149~150.



Appendix:

a. Solomon Data

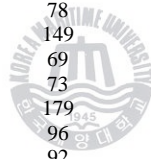
R101

VEHICLE
NUMBER CAPACITY
25 200

CUSTOMER
CUST NO. XCOORD. YCOORD. DEMAND READY TIME DUE DATE SERVICE TIME

| | | | | | | |
|----|----|----|----|-----|-----|----|
| 0 | 35 | 35 | 0 | 0 | 230 | 0 |
| 1 | 41 | 49 | 10 | 161 | 171 | 10 |
| 2 | 35 | 17 | 7 | 50 | 60 | 10 |
| 3 | 55 | 45 | 13 | 116 | 126 | 10 |
| 4 | 55 | 20 | 19 | 149 | 159 | 10 |
| 5 | 15 | 30 | 26 | 34 | 44 | 10 |
| 6 | 25 | 30 | 3 | 99 | 109 | 10 |
| 7 | 20 | 50 | 5 | 81 | 91 | 10 |
| 8 | 10 | 43 | 9 | 95 | 105 | 10 |
| 9 | 55 | 60 | 16 | 97 | 107 | 10 |
| 10 | 30 | 60 | 16 | 124 | 134 | 10 |
| 11 | 20 | 65 | 12 | 67 | 77 | 10 |
| 12 | 50 | 35 | 19 | 63 | 73 | 10 |
| 13 | 30 | 25 | 23 | 159 | 169 | 10 |
| 14 | 15 | 10 | 20 | 32 | 42 | 10 |
| 15 | 30 | 5 | 8 | 61 | 71 | 10 |
| 16 | 10 | 20 | 19 | 75 | 85 | 10 |
| 17 | 5 | 30 | 2 | 157 | 167 | 10 |
| 18 | 20 | 40 | 12 | 87 | 97 | 10 |
| 19 | 15 | 60 | 17 | 76 | 86 | 10 |
| 20 | 45 | 65 | 9 | 126 | 136 | 10 |
| 21 | 45 | 20 | 11 | 62 | 72 | 10 |
| 22 | 45 | 10 | 18 | 97 | 107 | 10 |
| 23 | 55 | 5 | 29 | 68 | 78 | 10 |
| 24 | 65 | 35 | 3 | 153 | 163 | 10 |
| 25 | 65 | 20 | 6 | 172 | 182 | 10 |
| 26 | 45 | 30 | 17 | 132 | 142 | 10 |
| 27 | 35 | 40 | 16 | 37 | 47 | 10 |
| 28 | 41 | 37 | 16 | 39 | 49 | 10 |
| 29 | 64 | 42 | 9 | 63 | 73 | 10 |
| 30 | 40 | 60 | 21 | 71 | 81 | 10 |
| 31 | 31 | 52 | 27 | 50 | 60 | 10 |
| 32 | 35 | 69 | 23 | 141 | 151 | 10 |
| 33 | 53 | 52 | 11 | 37 | 47 | 10 |
| 34 | 65 | 55 | 14 | 117 | 127 | 10 |
| 35 | 63 | 65 | 8 | 143 | 153 | 10 |
| 36 | 2 | 60 | 5 | 41 | 51 | 10 |
| 37 | 20 | 20 | 8 | 134 | 144 | 10 |
| 38 | 5 | 5 | 16 | 83 | 93 | 10 |
| 39 | 60 | 12 | 31 | 44 | 54 | 10 |
| 40 | 40 | 25 | 9 | 85 | 95 | 10 |
| 41 | 42 | 7 | 5 | 97 | 107 | 10 |
| 42 | 24 | 12 | 5 | 31 | 41 | 10 |
| 43 | 23 | 3 | 7 | 132 | 142 | 10 |

| | | | | | | |
|-----|----|----|----|-----|-----|----|
| 44 | 11 | 14 | 18 | 69 | 79 | 10 |
| 45 | 6 | 38 | 16 | 32 | 42 | 10 |
| 46 | 2 | 48 | 1 | 117 | 127 | 10 |
| 47 | 8 | 56 | 27 | 51 | 61 | 10 |
| 48 | 13 | 52 | 36 | 165 | 175 | 10 |
| 49 | 6 | 68 | 30 | 108 | 118 | 10 |
| 50 | 47 | 47 | 13 | 124 | 134 | 10 |
| 51 | 49 | 58 | 10 | 88 | 98 | 10 |
| 52 | 27 | 43 | 9 | 52 | 62 | 10 |
| 53 | 37 | 31 | 14 | 95 | 105 | 10 |
| 54 | 57 | 29 | 18 | 140 | 150 | 10 |
| 55 | 63 | 23 | 2 | 136 | 146 | 10 |
| 56 | 53 | 12 | 6 | 130 | 140 | 10 |
| 57 | 32 | 12 | 7 | 101 | 111 | 10 |
| 58 | 36 | 26 | 18 | 200 | 210 | 10 |
| 59 | 21 | 24 | 28 | 18 | 28 | 10 |
| 60 | 17 | 34 | 3 | 162 | 172 | 10 |
| 61 | 12 | 24 | 13 | 76 | 86 | 10 |
| 62 | 24 | 58 | 19 | 58 | 68 | 10 |
| 63 | 27 | 69 | 10 | 34 | 44 | 10 |
| 64 | 15 | 77 | 9 | 73 | 83 | 10 |
| 65 | 62 | 77 | 20 | 51 | 61 | 10 |
| 66 | 49 | 73 | 25 | 127 | 137 | 10 |
| 67 | 67 | 5 | 25 | 83 | 93 | 10 |
| 68 | 56 | 39 | 36 | 142 | 152 | 10 |
| 69 | 37 | 47 | 6 | 50 | 60 | 10 |
| 70 | 37 | 56 | 5 | 182 | 192 | 10 |
| 71 | 57 | 68 | 15 | 77 | 87 | 10 |
| 72 | 47 | 16 | 25 | 35 | 45 | 10 |
| 73 | 44 | 17 | 9 | 78 | 88 | 10 |
| 74 | 46 | 13 | 8 | 149 | 159 | 10 |
| 75 | 49 | 11 | 18 | 69 | 79 | 10 |
| 76 | 49 | 42 | 13 | 73 | 83 | 10 |
| 77 | 53 | 43 | 14 | 179 | 189 | 10 |
| 78 | 61 | 52 | 3 | 96 | 106 | 10 |
| 79 | 57 | 48 | 23 | 92 | 102 | 10 |
| 80 | 56 | 37 | 6 | 182 | 192 | 10 |
| 81 | 55 | 54 | 26 | 94 | 104 | 10 |
| 82 | 15 | 47 | 16 | 55 | 65 | 10 |
| 83 | 14 | 37 | 11 | 44 | 54 | 10 |
| 84 | 11 | 31 | 7 | 101 | 111 | 10 |
| 85 | 16 | 22 | 41 | 91 | 101 | 10 |
| 86 | 4 | 18 | 35 | 94 | 104 | 10 |
| 87 | 28 | 18 | 26 | 93 | 103 | 10 |
| 88 | 26 | 52 | 9 | 74 | 84 | 10 |
| 89 | 26 | 35 | 15 | 176 | 186 | 10 |
| 90 | 31 | 67 | 3 | 95 | 105 | 10 |
| 91 | 15 | 19 | 1 | 160 | 170 | 10 |
| 92 | 22 | 22 | 2 | 18 | 28 | 10 |
| 93 | 18 | 24 | 22 | 188 | 198 | 10 |
| 94 | 26 | 27 | 27 | 100 | 110 | 10 |
| 95 | 25 | 24 | 20 | 39 | 49 | 10 |
| 96 | 22 | 27 | 11 | 135 | 145 | 10 |
| 97 | 25 | 21 | 12 | 133 | 143 | 10 |
| 98 | 19 | 21 | 10 | 58 | 68 | 10 |
| 99 | 20 | 26 | 9 | 83 | 93 | 10 |
| 100 | 18 | 18 | 17 | 185 | 195 | 10 |



b. MATLAB Codes

```

%=====
%                               Coded by Tran Ngoc Hoang Son                               %
%                               Korea Maritime University                               %
%                               Department of Logistics                               %
%                               2009                                                 %
%=====
%-----
function newSERVE=decranking(CUSLIST, SERVE, a, v, gamma)
    tempSERVE=SERVE;
    newSERVE=pluckingoff(CUSLIST, tempSERVE, a, v, gamma);

    while sumsqr(tempSERVE~=newSERVE)>0
        tempSERVE=newSERVE;
        newSERVE=pluckingoff(CUSLIST, tempSERVE, a, v, gamma);
    end
    newSERVE;
end
%-----
function newSERVE=pluckingoff(CUSLIST, SERVE, a, v, gamma)
    newSERVE=SERVE;
    [r c]=size(SERVE);
    for i=1:c
        u=2;v=3;
        while u<r
            for v=u+1:r-1
                tempSERVE=newSERVE;
                s=u;t=v;
                while s<t
                    tt=tempSERVE(s,i);
                    tempSERVE(s,i)=tempSERVE(t,i);
                    tempSERVE(t,i)=tt;
                    s=s+1;t=t-1;
                end
                [newtime
newcost]=calculatetimeandcost4(CUSLIST, newSERVE, a, v, gamma);
                [temptime
tempcost]=calculatetimeandcost4(CUSLIST, tempSERVE, a, v, gamma);
                if tempcost(i)<newcost(i)
                    newSERVE=tempSERVE;
                end
            end
            u=u+1;
        end
    end
end
%-----

```



```

%Nearest L-Neighbor Method
function [SERVE]=multivehiclerouting4(CUSLIST,a,v,k,L)
    UNSERVE=CUSLIST(2:length(CUSLIST),1)';
    SERVE=zeros(1,k);indexes=ones(1,k);
    len=length(UNSERVE);
    kth=1;
    while len>0
        if len>L
            [list time cost remain]=selectroutes5(CUSLIST, UNSERVE,
SERVE(indexes(kth),kth),a,v, L);
        else
            [list time cost remain]=selectroutes5(CUSLIST, UNSERVE,
SERVE(indexes(kth),kth),a,v, len);
        end

        SERVE=[SERVE;zeros(length(list),k)];
        SERVE(indexes(kth)+1:indexes(kth)+length(list),kth)=list;
        indexes(kth)=indexes(kth)+length(list);
        for u=1:k
            if u==kth
                continue;
            end
            SERVE(indexes(u)+1:indexes(kth),u)=[SERVE(indexes(u),u)*ones(index
es(kth)-indexes(u),1)];
            indexes(u)=indexes(kth);
        end
        UNSERVE=remain;
        len=length(UNSERVE);
        kth= mod(kth,k)+1;
    end
end
%-----
function [list time cost remain]=selectroutes5(CUSLIST, UNSERV,
cuscode, a,v,L)
    [row,col]=size(CUSLIST);
    %---initialize---
    serve_time=0;
    serve_cost=0;

    x=2;
    gamma=[];

    len=length(UNSERV);

    if (len<L)
        % we can control this case by using this function with len
    end

    len=length(UNSERV);

```

```

mode=0;%dynamic travelling time mode =1
[gammanextlist x]=gammagenerator(x,L,mode);
gamma=[gamma gammanextlist];

index=[1:len];

SERVE=[cuscode];
k=1;
r=L;
seq=index(1:r);
seq_index=[1:length(seq)];

SERVE_temp=addlist(SERVE,getvaluefromsequence(UNSERV,getvaluefroms
equence(seq,seq_index)),k,L);
% a=[1 0 0];v=1;
[time
cost]=calculatedtimeandcost4(CUSLIST,[SERVE_temp],a,v,[gamma]);

OPTLIST=SERVE_temp;
optlistcost=cost;
optlisttime=time;
optseq=seq;%should store for eliminating the served customers

ff=1;
while ff==1
    seq_index=[1:length(seq)];

    f=1;
    while f==1
SERVE_temp=addlist(SERVE,getvaluefromsequence(UNSERV,getvaluefroms
equence(seq,seq_index)),k,L);
        [time
cost]=calculatedtimeandcost4(CUSLIST,[SERVE_temp],a,v,[gamma]);
        if sum(optlistcost)>sum(cost)
            OPTLIST=SERVE_temp;
            optlistcost=cost;
            optlisttime=time;
            optseq=seq;
        end

        [seq_index f]=permutation_next(seq_index);
    end
    [seq ff]=next_r_combination(seq,len);
end

%update SERVE list
SERVE=OPTLIST;

```



```

%eliminate the served list in UNSERV
for i=1:length(optseq)
    UNSERV(optseq(i)-i+1)=[];
end

remain=UNSERV;

list=SERVE(2:length(SERVE));

[time
cost]=calculatetimeandcost4(CUSLIST,[SERVE],a,v,[gamma]);

end
%-----
function serve_next=addlist(serve,seqlist,k,L)
    serve_next=serve;
    for u=1:L
        serve_next=[serve_next;seqlist((u-1)*k+1:u*k)];%[x1y1z1
x2y2z2 xyz]
    end
end
%-----
function [gamma x]=gammagenerator(xo,step,mode)
    gamma_array=[-0.5:0.5:8];%planned speed
    if mode==0%planned speed
        x=2;
        for u=1:step
            gamma(u)=gamma_array(x);
        end
        return
    end
    m=17;
    a=3;
    x=xo;
    %gamma(1)=mod((a*x0),m);
    for u=1:step
        %x=mod((alpha*x),m);%case of dynamic travel time
        %x=2;%case of planned speed
        x=mod(a*x,m);
        gamma(u)=gamma_array(x);
    end
end
%-----
function s=getvaluefromsequence(seq,seqindex)
    for k=1:length(seqindex)
        s(k)=seq(seqindex(k));
    end
    %return s;
end

```

```

%-----
function [time
cost]=calculatetimeandcost4(cuslist,serve,a,v,gamma)
%a=[1 1 1 1]; %weighting parameters
%V=1;% assume average velocity of vehicle, planned speed
%t=s/v, T=Tij(1+gamma), -0.5<=gamma<=inf. gamma=-0.5: good
conditions, gamma=inf: bad conditions on road

    [row col]=size(serve);%rows contain sequence of served
customers
    cost=zeros(1,col);
    time=zeros(1,col);
    for u=2:row %along customers sequence
        for v=1:col %for each vehicle
            arrivetime=v\norm([(cuslist(serve(u,v)+1,2)-
cuslist(serve(u-1,v)+1,2)) ...
(cuslist(serve(u,v)+1,3)-cuslist(serve(u-
1,v)+1,3))])*(1+gamma(u-1));%+variant time gamma
            %update x at each link->need to save sequence of x's
values for
            %comparasion between
            waitingtime=max( 0, cuslist(serve(u,v)+1,5)-
(time(v)+arrivetime) );
            servicetime=cuslist(serve(u,v)+1,7);
            time(v)=time(v)+arrivetime+waitingtime+servicetime;
            delaytime=max(0, time(v)-cuslist(serve(u,v)+1,6));

            cost(v)=cost(v)+...
                a(1)*arrivetime+...
                a(2)*waitingtime+...
                a(3)*delaytime;
        end
    end
end
end

%-----
%find the next r combination
function [pnext f]=next_r_combination(p,n)
    l=length(p);

    pnext=n-l+1:n;
    if pnext==p
        f=0;
        return
    end

    k=1;
    while p(k)==n-l+k
        k=k-1;

```

```

end
p(k)=p(k)+1;
for s=k+1:l
    p(s)=p(k)+s-k;
end
f=1;
pnext=p;
end
%-----
%find the next permutation
function [pnext f]=permutation_next(p)
    n=length(p);
    if (p==sort([1:n], 'descend')) %check last permutation n,n-
1,...,1
        f=0;
        pnext=p;
        return;
    end

    pcheck=sort(p);
    if (pcheck~= [1:n]) %check last permutation n,n-1,...,1
        f=0;
        pnext=p;
        return;
    end

    pnext=p;

    j=length(p)-1;
    while ((p(j)>p(j+1)) && (j>0) )
        j=j-1;
    end

    if j==0
        j=1;
    end

    k=n;
    while p(j)>p(k)
        k=k-1;
    end

    %swap(j,k);
    t=p(j);
    p(j)=p(k);
    p(k)=t;

    k=n;

```



```

m=j+1;
while k>m
    %swap(k,m);
    t=p(k);
    p(k)=p(m);
    p(m)=t;

    k=k-1;
    m=m+1;
end

pnext=p;
f=1;
end
%-----
%find the previous permutation
function [pnext f]=permutation_prev(p)
    n=length(p);
    if (p==[1:n]) %check last permutation n,n-1,...,1
        f=0;
        pnext=p;
        return;
    end

    pcheck=sort(p);
    if (pcheck~= [1:n]) %check last permutation n,n-1,...,1
        f=0;
        pnext=p;
        return;
    end

    pnext=p;

    j=length(p)-1;
    while ((p(j)<p(j+1)) && (j>0) )
        j=j-1;
    end

    if j==0
        j=1;
    end

    k=n;
    while p(j)<p(k)
        k=k-1;
    end

    %swap(j,k);

```

```
t=p(j);
p(j)=p(k);
p(k)=t;

k=n;
m=j+1;
while k>m
    %swap(k,m);
    t=p(k);
    p(k)=p(m);
    p(m)=t;

    k=k-1;
    m=m+1;
end

pnext=p;
f=1;
end
```



Bibliography

- [1] Martello, S., Laporte, G., Minoux, M. and Ribeiro, C.(1987), "Surveys in Combinatorial Optimization", Annals of Discrete Mathematics, Mathematics Studies, North-Holland, Vol. 132, Chapter 5 and 9.
- [2] Ball, M., Magnanti, T., Monma, C. and Nemhauser, G.(1995)a, "Handbooks in Operation Research and Management Science – Network Model", Elsevier Science, Vol.7.
- [3] Ball, M., Magnanti, T., Monma, C. and Nemhauser, G.(1995)b, "Handbooks in Operation Research and Management Science – Network Routing", Elsevier Science, Vol.8.
- [4] Horowitz, E. and Sahni, S.(1978), "Fundamentals of Computer Algorithms", Computer Science Press.
- [5] Horowitz, E. and Sahni, S.(1983), "Fundamentals of Data Structures", Computer Science Press.
- [6] Syslo, M.M., Deo, N. and Kowalik J.S.(1983), "Discrete Optimization Algorithms with Pascal Programs", Dover Edition.
- [7] Ahuja, R.K., Magnanti, T.L. and Orlin J.B.(1993), "Network flows: theory, algorithms and applications", 1st edition, Prentice Hall.
- [8] Gendreau, M., Laporte G. and Seguin R.(1996), "Stochastic vehicle routing", European Journal Of Operational Research, Vol. 88, pp. 3~12.
- [9] Michalewicz (1996), "Genetic algorithms + data structures = evolution programs", 3rd edition, Springer.

- [10] Sipser, M.(1997), "Introduction to the Theory of Computation", PWS Publishing.
- [11] Toth, P. and Vigo, D.(2002), "The Vehicle Routing Problem", Society for Industrial and Applied Mathematics (SIAM).
- [12] Ichoua, S., Gendreau, M. and Potvin, J.Y.(2003), "Vehicle Dispatching with Time-Dependent Travel Times", European Journal of Operational Research, Vol. 114, pp. 379~396.
- [13] Montemanni, R. Gambardella, L.M., Rizzoli, A.E. and Donati, A.V.(2005), "Ant colony system for a dynamic vehicle routing problem", Journal of Combinatorial Optimization, Vol. 10, pp. 327~343.
- [14] Haghani, A. and Jung, S.(2005), "A Dynamic Vehicle Routing Problem with Time-Dependent Travel Times", Journal of Computers & Operations Research, Vol. 32, pp. 2959~2986.
- [15] Hashimoto, H., Ibaraki, T., Imahori, S. and Yagiura, M.(2006), "The Vehicle Routing Problem with Flexible Time Windows and Travelling Times", Journal of Discrete Applied Mathematics, Vol. 154, pp. 2271~2290.
- [16] Fabri, A. and Recht, P.(2006), "On Dynamic Pickup and Delivery Vehicle Routing with Several Time Windows and Waiting Times", Journal of Transportation Research Part B, Vol. 40, pp. 335~350.
- [17] Hanshar, F.T. and Ombuki-Berman, B.M.(2007), "Dynamic vehicle routing using genetic algorithms", Applied Intelligent, Vol. 27, pp. 89~99.

- [18] Chitty, D.M. and Hernandez, M.L.(2004), “A hybrid ant colony optimization technique for dynamic vehicle routing”, Book chapter, Genetic and Evolutionary Computation – GECCO 2004, pp. 48-59.
- [19] Zeimpekis, V. and Giaglis, G.M.(2005), “A Dynamic Real-Time Vehicle Routing System for Distribution Operations”, Consumer Driven Electronic Transformation, Springer.
- [20] Fan, J., Wang, X. and Ning, H.(2006), “A Multiple Vehicles Routing Problem Algorithm with Stochastic Demand”, The Sixth World on Intelligent Control and Automation WCICA.
- [21] Simchi-Levi, D., Kaminsky, P. and Simchi-Levi, E.(2003), “Design and Managing the Supply Chain: Concepts, Strategies and Case Studies”, McGraw Hill.
- [22] Solomon data, <http://www.idsia.ch/~luca/macsvrptw/problems/welcome.htm>
- [23] Shin, J.H., Chang, M.H., and Yu, S.J.(2008), “A Study on RTLS Technology based YT Dynamic Operation for Efficiency of Container Terminal”, Journal of Navigation and Port Research, Vol. 32, No. 5, pp. 369-377.
- [24] Shin, J.H., Yu, Yu, S.J., Chang, M.H.(2008), “A Study on Optimized Decision Model for Transfer Crane Operation in Container Terminal”, Journal of Navigation and Port Research, Vol. 32, No. 6, pp. 465-471.
- [25] Shin, J.Y. and Oh, S.I.(2008), “Vehicle Routing Problem for delivering containers using combined chassis trailers”, Conference of KINPR 2008, pp. 155-156.

- [26] Arya, S., Mount, D. M., Netanyahu, N. S., Silverman, R. and Wu, A. Y.(1994), “An Optimal Algorithm for Approximate Nearest Neighbor Searching in Fixed Dimensions”, Journal of the ACM, Vol. 45, No. 6, pp. 891-923.
- [27] Russell, S., and Norvig, P.(2003), “Artificial Intelligence: A Modern Approach”, Prentice Hall, American.
- [28] Michalewicz, Z., and Fogel, D.(2004), “How to Solve It: Modern Heuristics”, Second, Revised and Extended Edition, Springer, Germany.
- [29] Larsen, A.(2000), The Dynamic Vehicle Routing Problem. PhD dissertation, Technical University of Denmark.
- [30] Ichoua, S., Gendreau, M. and Potvin, J.V.(2000),), “Diversion Issues in Real-Time Vehicle Dishpatching”, Transportation Science, Vol. 34, No. 4, pp. 426~438.
- [31] Berman, O. Larson, R.C.(2001), “Deliveries in an Inventory/Routing Problem Using Stochastic Dynamic Programming”, Transportation Science, Vol. 35, No. 2, pp. 192~213.
- [32] Kenyon, A.S. and Morton, D.P.(2003), “Stochastic Vehicle Routing with Random Travel Times”, Transportation Science, Vol. 37, No. 1, pp. 69~82.
- [33] Taniguchi, E. and Shimamoto, H.(2004), “Intelligent Transportation System Based Dynamic Vehicle Routing and Scheduling with Variable Travel Times”, Transportation Research – Part C, Vol. 12, No. 3 – 4 , pp. 235~250.
- [34] Branke, J., Midendorf, M., Noeth, G. and Dessouky, M.(2005), “Waiting Strategies for Dynamic Vehicle Routing”, Transportation Science, Vol. 39, No. 3, pp. 298~312.

- [35] Hvattum, L.M., Løkketangen, A. and Laporte, G.(2006), “Solving a Dynamic and Stochastic Vehicle Routing Problem with a Sample Scenario Hedging Heuristic”, *Transportation Science*, Vol. 40, No. 4, pp. 421~438.
- [36] Hvattum, L.M., Løkketangen, A. and Laporte, G.(2007), “A Branch-and-Regret Heuristic for Stochastic and Dynamic Vehicle Routing Problems”, *Wiley Periodicals of Networks*, Vol. 49, No. 4, pp. 330~340.
- [37] Ghiani, G., Guerriero, F., Laporte, G., Musmanno, R.(2003), “Real-time Vehicle Routing: Solution Concepts, Algorithms and Parallel Computing Strategies”, *European Journal of Operational Research*, Vol. 151, pp. 1~11.
- [38] Hamming, R.(1991), “The Art of Probability”, Addition Wesley.
- [39] Sauer, T.(2006), “Numerical Analysis”, Addition Wesley.



My wonders of love and life:

Lost-found-named my Unkyon (은견), is that destiny?

-No, that is not, that is just a causal progress!

Do grief and torment blend with recognized long time waiting love?

-Yes, a lot! Astonishing emotion of love!

Is that really happy or just a sensation stage people would pass?

-Maybe both!

What is a PhD?

Plagiarist will never be a PhD! If were, shame on you!

Just original self-finding one could be...



Cảm ơn ba má...