CHAPTER V

SINGLE ITEM-OUTLET HEURISTIC (SIOH)

5.1 Introduction

This chapter presents a heuristic for the multi-item multi-depot inventory routing problem as proposed in chapter III. This heuristic is modified from the heuristic which is presented in the chapter IV. Since the result in chapter IV show that the heuristic which is proposed in that chapter can efficiently solve the problem with single-item single-depot, this chapter extends that heuristic to solve problem with multi-item multi-depot.

The remainder of this chapter is organized as follows. Modification of heuristic is presented in Section 5.2. A performance measurement will be presented in Section 5.3. The computational results are reported in Section 5.4. The conclusions of this work and some recommendations for the next are presented in Section 5.5.

5.2 Heuristic description

5.2.1 Modification of heuristic to solve multi-item multi-depot problem

We first consider problem with multi-item by transforming it into problem with single-item. Transformation approach is to consider each item on each outlet as an item-outlet. Now each outlet will be considered as item-outlet with a number of items. For instance the problem with two items, each outlet will be considered as two item-outlets. The illustration of the transformation of outlets and items into item-outlets is shown in Figure 5.1.

Like the heuristic in chapter III, the heuristic procedure will start from initial solution by Lot-For-Lot but the route and replenishment quantity is decided for item-outlets instead outlets.

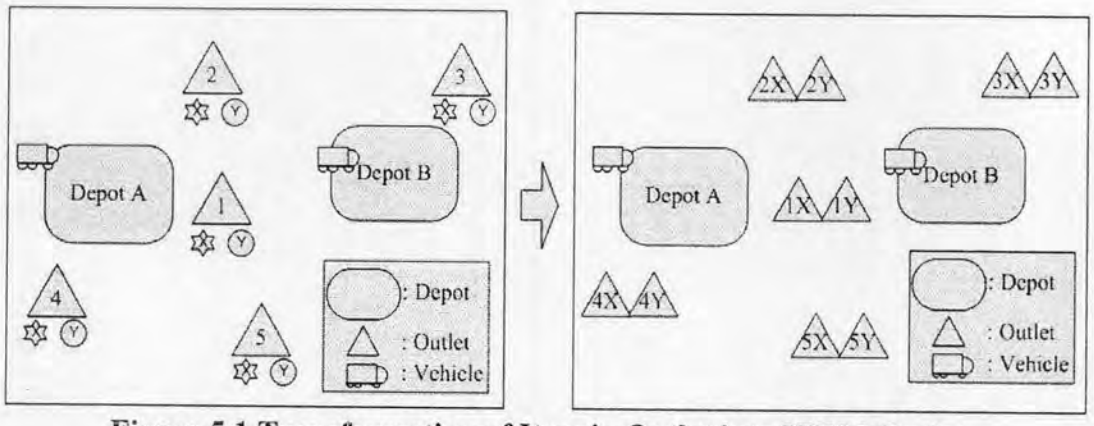


Figure 5.1 Transformation of Item in Outlet into ITEM-Outlet

The Step 2 of SIOH differs from the heuristic in chapter IV in that SIOH will randomly select an item-outlet while the heuristic in chapter IV will select an outlet. Another modification is the shortest path network. While the simple problem does not include vehicle carrying capacity and outlet storage capacity, the multi-item multi-depot problem does. This causes the shortest path network to be modified. The modification occurs at arc building process, the arc will exist if and only if the replenishment quantity according to that arc does not violate both vehicle carrying capacity and outlet storage capacity. So the step 3.1 in this chapter will create the shortest path networks for every item-outlet instead every outlet while each period is still represented by a node and each replenishment period is represented by an arc. Now an arc from k to t means considered item-outlet inventory level is replenished at period k and then period t. These networks are solved by an efficient shortest path algorithm in order to determine delivery period of each item-outlet.

One significant condition is that all arcs in the shortest path network must not violate both vehicle carrying capacity and outlet storage capacity. For vehicle carrying capacity, the arcs must be checked that they will not violate vehicle carrying capacity by pre-assigning them into an origin period of arc. For example, an arc "2-4" of the item-outlet named "1X" which means there are deliveries at period 2 and then period 4. This arc must be checked by re decision about route in period 2

with addition information of replenishment quantity of this item-outlet with equal to cumulative demand of this item-outlet from period 2 to period 3. Moreover, essentially, the new route will be acceptable as a feasible route if carrying capacity is not exceeded. And if the new route in origin period is feasible, this arc will be considered as an available arc. The illustration of the case that vehicle capacity is met is shown in Figure 5.2 and the case that vehicle capacity is violated in shown in Figure 5.3. In the Figure 5.2, the arc 1-3 of item-outlet "1X" is considered by using summation of demand from period 1 to period 2 as replenishment quantity of "1X" in period 1. This amount of replenishment quantity is considered as delivery load in routing decision. In the Figure 5.3, the arc 1-4 of item-outlet "1X" is considered and results in violation of vehicle capacity.

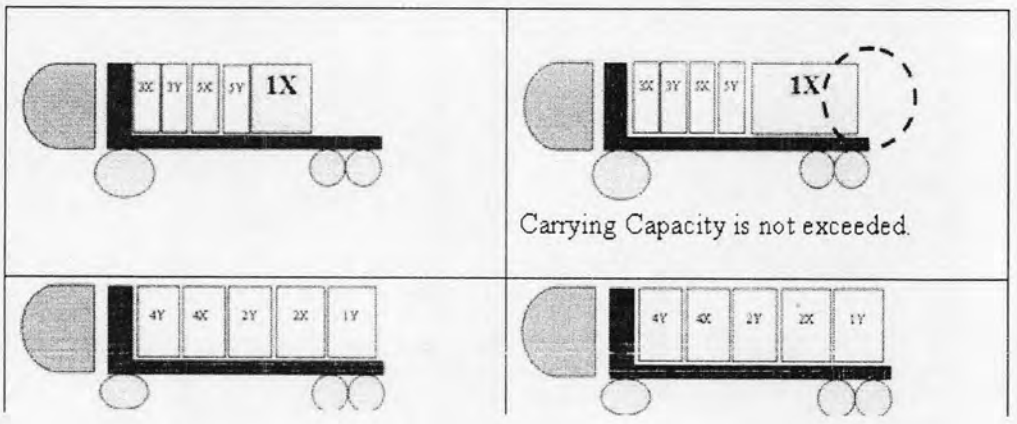


Figure 5.2 Illustration of the case that vehicle capacity is met

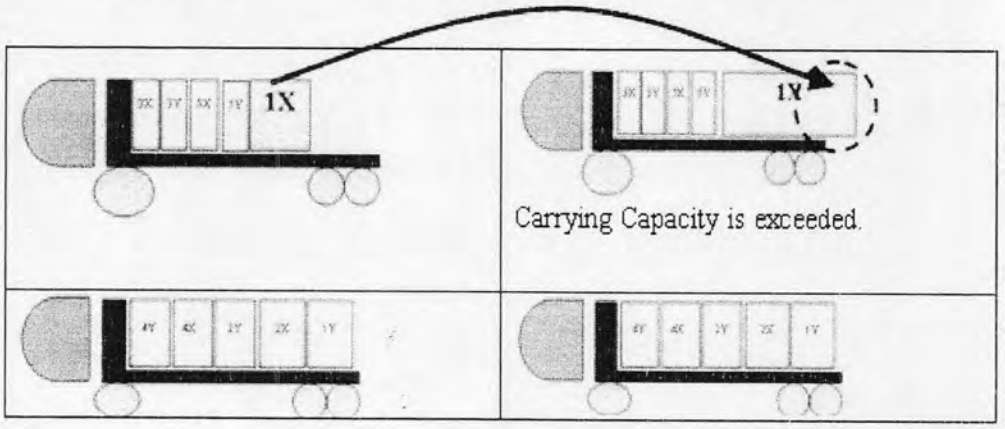


Figure 5.3 Illustration of the case that vehicle capacity is violated

The other significant factor is the outlet storage capacity. The outlet must be checked also by adding replenishment quantity of this arc into existing

replenishment quantity of other outlets locating in the same outlet. As the constraint, the total replenishment items of adding and existing replenishments must not exceed the outlet storage capacity. The illustration of the case that storage capacity is met is shown in Figure 5.4 and the case that storage capacity is violated in shown in Figure 5.5. In the Figure 5.4, the arc 1-3 of item-outlet "1X" is considered by using summation of demand from period 1 to period 2 as replenishment quantity of "1X" in period 1. This amount of replenishment quantity must not cause the storage capacity exceeded. In the Figure 5.5, the arc 1-4 of item-outlet "1X" is considered and results in violation of storage capacity.

The last modification is the routing algorithm instead of consider route for a vehicle now this heuristic has to use routing algorithm to allocate item-outlets and route them simultaneously due to there is vehicle carrying capacity so a vehicle may be not enough for delivery service and can affect to transportation cost in term of both routing cost and vehicle fixed-charge cost.

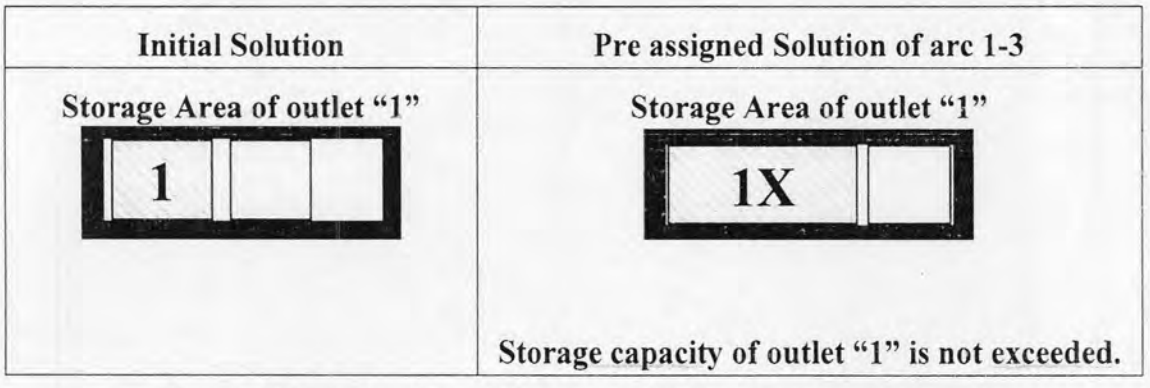


Figure 5.4 Illustration of the case that storage capacity is met

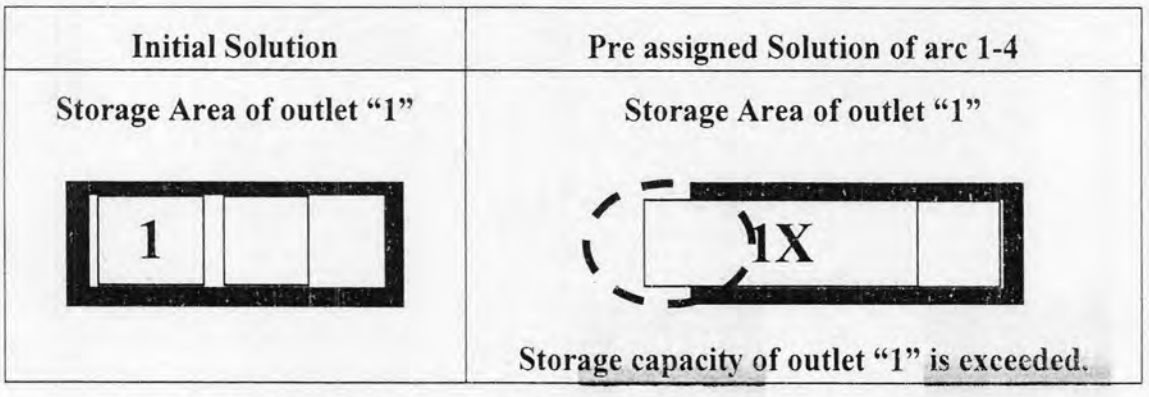


Figure 5.5 Illustration of the case that storage capacity is violated

5.2.2 Numerical example

In this section, a simple problem is illustrated. This example consists of two depots and five outlets with two items in each outlet. Each depot has a vehicle for delivery items to outlets. All vehicles have a capacity limit at one hundred units Decisions are made for three periods. All outlets are transformed into item-outlets and shown in Figure 5.6. Figure 5.6 illustrates the graphical location of all depots and item-outlets in the example problem. Table 5.1 shows the distance between all locations. Demand of all outlets is given in Table 5.2. Table 5.3 gives the information about the holding cost of the outlets. Beginning inventory of all outlets are assumed to be zero.

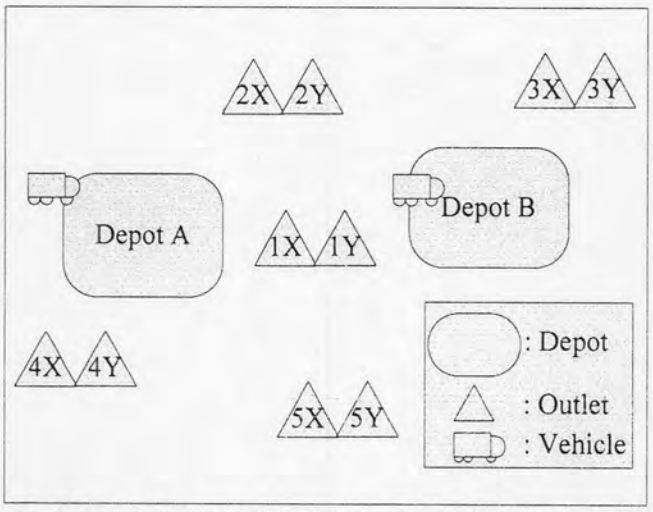


Figure 5.6 Illustration of the graphical location of all depots and outlets in the example problem.

Table 5.1 Distance between all locations

		Depot		Outlet				
		A	**B**	**1**	**2**	**3**	**4**	**5**
Depot	**A**	0	3	7	10	9	9	11
	B	3	0	9	12	7	7	13
Outlet	**1**	9	7	0	13	16	11	16
	2	12	10	13	0	15	19	4
	3	7	9	16	15	0	11	13
	4	7	9	11	19	11	0	20
	5	13	11	16	4	13	20	0

Table 5.2 Demand of all outlets

		Item \ Period	1	2	3
Outlet	1	X	20	19	20
		Y	19	20	19
	2	X	20	21	20
		Y	17	20	21
	3	X	21	22	20
		Y	24	20	15
	4	X	21	18	25
		Y	20	20	21
	5	X	16	20	23
		Y	22	22	17

Table 5.3 Holding cost of all outlets

		Item	Holding Cost
Outlet	1	X	0.10
		Y	0.15
	2	X	0.25
		Y	0.30
	3	X	0.07
		Y	0.10
	4	X	0.35
		Y	0.40
	5	X	0.14
		Y	0.17

Step 1: Set up an initial phase

The example of initial solution is presented in Table 5.4 and Figure 5.7. Table 5.4 presents the replenishment quantity of each item-outlet, for example item-outlet "1X" will receive replenishment quantity 12 units at period 1, 8

units at period 2 and 10 units at period 3. There are two delivery routes for all periods. The first route starts from Depot "B" then visit "5Y", "5X", "1X", "3X", "3Y" and return to Depot "B". The second route starts from Depot "A" then visit "4X", "4Y", "1Y", "2Y", "2X" and return to Depot "A".

Table 5.4 Replenishment quantity and delivery route of initial solution

Period / Item-Outlet	1	2	3
1X	12	8	10
1Y	11	9	10
2X	10	8	9
2Y	8	10	11
3X	12	8	10
3Y	11	9	10
4X	10	8	9
4Y	8	10	11
5X	10	8	9
5Y	8	10	11
Route1	B-5Y-5X-1X-3X-3Y-B	B-5Y-5X-1X-3X-3Y-B	B-5Y-5X-1X-3X-3Y-B
Route2	A-4X-4Y-1Y-2Y-2X-A	A-4X-4Y-1Y-2Y-2X-A	A-4X-4Y-1Y-2Y-2X-A

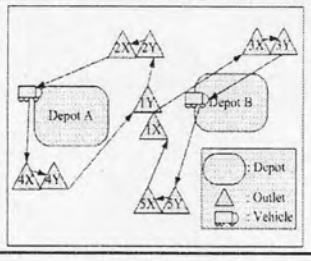
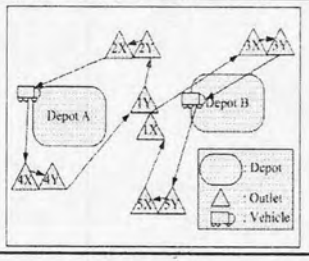
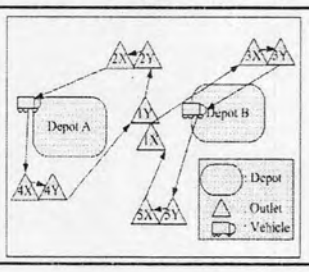
Period	1	2	3
Route			

Figure 5.7 Initial route

Step 2: select an outlet

The illustration of selection of an item-outlet is shown in Figure 5.8. In Figure 5.8, an item-outlet named "1X" is selected to be considered for

determining replenishment quantity of it. The shortest path network is used to determine the replenishment quantity.

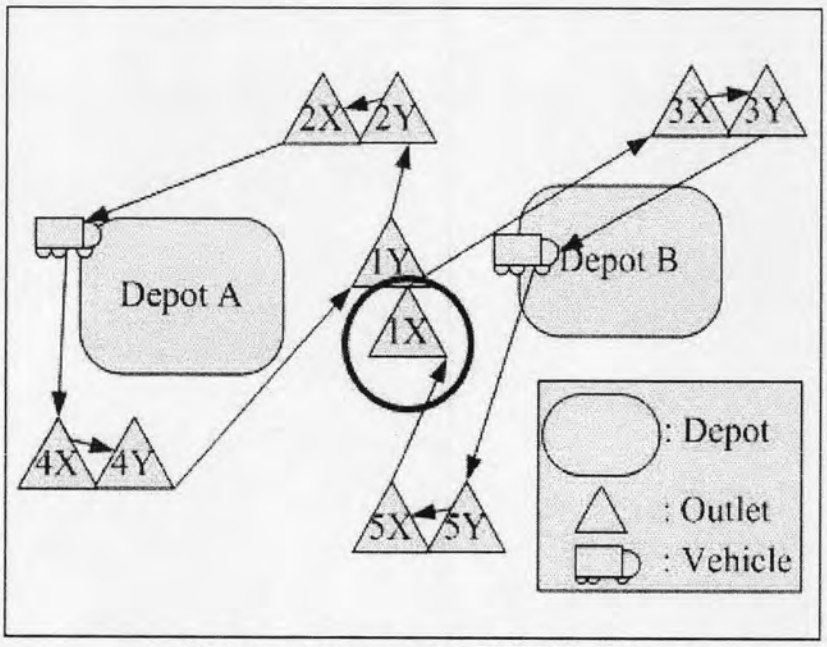


Figure 5.8 Item-outlet "1X" is selected

Step 3: Determine delivery period using shortest path algorithm.

The Lot-Sizing Problem of item-outlet "1X" is transformed into the shortest path problem and solved by using efficient shortest path algorithm. The Lot-Sizing Problem of the selected item-outlet is solved by assuming all routes and all other item-outlets replenishment quantity are as same as the solution in previous iteration. The information that will be used in the shortest path problem composes of demand of the selected item-outlet, all current routes and replenishment quantity of other item-outlets. The Lot-Sizing Problem of item-outlet "1X" can be considered as the Lot-Sizing Problem of item "X" of outlet "1". The result of the shortest path problem is the delivery period that it will be replenished.

Step 3.1 Define Node and Arc.

The illustration of shortest path network is presented in the Figure 5.9.

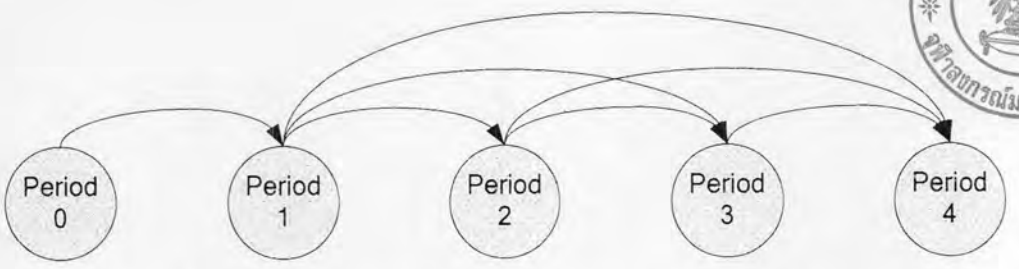


Figure 5.9 Illustration of shortest path network

In figure 5.9, each node stands for node, period 0, 1, 2, ..., 4 and each arrow line stands for arc, replenishment period 0-1, 1-2,..., 3-4. Hence, we will name node 0, node 1, ..., and node 4, and arc 0-1, arc 1-2, arc 1-3, ..., and arc 3-4. The meaning of arc k-t is there is replenishment at period t after replenishment at period k. For example arc 2-3 means the item-outlet "1X" is replenished at period 3 after replenishment at period 2. For the case of arc 0-t means the item-outlet "1X" is first replenished at period t, e.g. an arc 0-1 means the first replenishment to the item-outlet "1X" is made at period 1. For the case of arc k-4, which node 4 is the last node in this problem, means the item-outlet "1X" is last replenished at period k, e.g. an arc 2-4 means the last replenishment to the item-outlet "1X" is made at period 2 to cover demand of period 2 to period 3. It can be notice that there are no arc 0-2, 0-3 and 0-4 because these arcs violate the stock out constraint. For example arc 0-2 violates the stock out constraint since the first replenishment is made at period 2 which the demand at period 1 is not been satisfied.

Step 3.2 Define Arc cost.

Cost of every arc is the summation of increasing holding cost and the difference of route cost of selecting the considered arc. An increasing holding cost of arc k-t is determined by sum all inventory holding cost of the considered outlet from period t to period k+1. Rout cost is determined by calculating the difference between route cost including considered this outlet and route cost without considered this outlet at period k. The illustration of arc cost of shortest path network of the item-outlet "1X" is shown in Figure 5.11.

Table 5.5 Holding Cost of Replenishment Period following Arc

From Period \ To Period	1	2	3	4
0	0	-	-	-
1	-	0	8x0.1=0.8	(18+10)x0.1=2.8
2	-	-	0	10x0.1=1
3	-	-	-	0

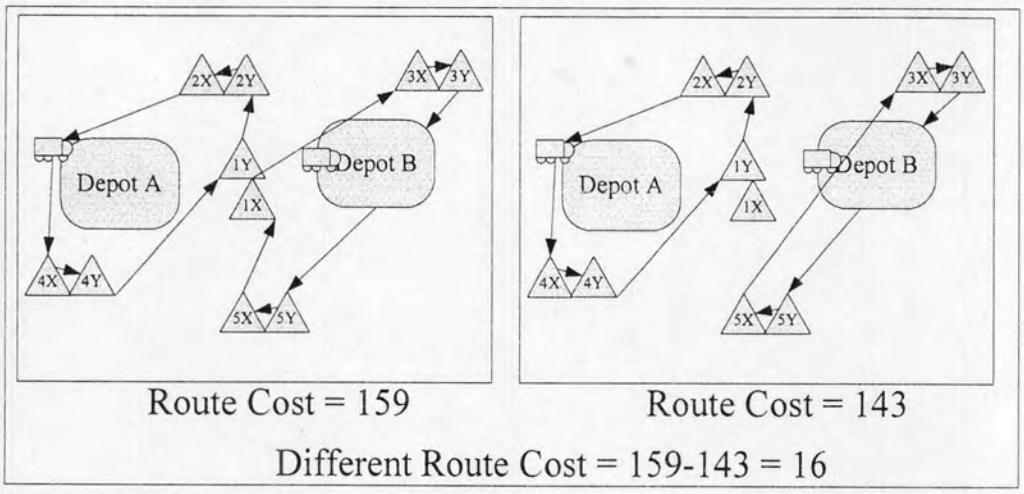


Figure 5.10 Illustration of different route cost in period 2

Table 5.6 Arc Cost of "1X"

From Period \ To Period	1	2	3	4
0	0	-	-	-
1	-	0+16=16	0.8+16=16.8	2.8+16=18.8
2	-	-	0+16=16	1+16=17
3	-	-	-	0+16=16

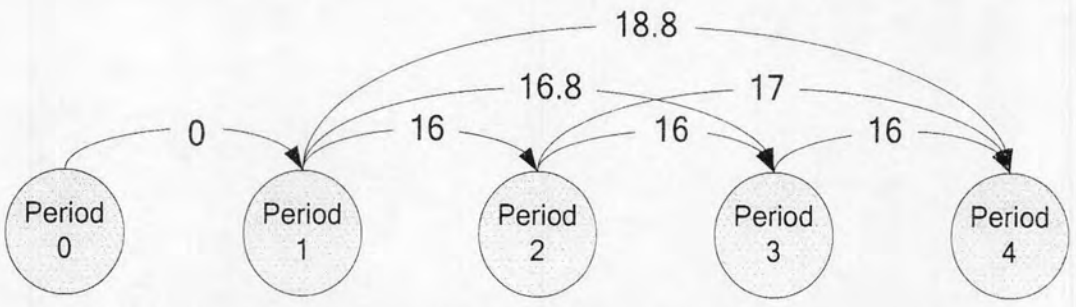


Figure 5.11 Illustration of arc cost of shortest path network

Step 3.3 and 3.4 Determine decision variables using results from the shortest path algorithm.

The solution of shortest path problem of the item-outlet "1X" in this iteration is 0-1-4 so there is delivery schedule for "1" on period 1 only.

Step 4: Update replenishment quantity and route

The solution after updating is shown in Table 5.7 and Figure 5.12. In Table 5.7, the replenishment quantity is presented. It can be indicated that the replenishment quantity of the outlet "1X" in period 1, 2 and 3 are changed. The value at period 1 is 30 which is derived from summation of demand from period 1 to period 3. The period 2 and 3 has no replenishment quantity since the replenishment quantity in period 1 can cover demand in this period already. In Figure 5.12 shows the illustration of delivery route in each period. The routes in period 2 and 3 are changed from previous state. The first route in period 2 and 3 becomes B-5X-5Y-3X-3Y-B that means the vehicle starts from depot indexed by "B" and goes to item-outlet "5X", "5Y", "3X", "3Y" and return to "B".

Table 5.7 Replenishment quantity and delivery route after updating

Period / Item-Outlet	1	2	3
1X	12+8+10=30	0	0
1Y	11	9	10
2X	10	8	9
2Y	8	10	11
3X	12	8	10
3Y	11	9	10
4X	10	8	9
4Y	8	10	11
5X	10	8	9
5Y	8	10	11
Route1	B-5Y-5X-1X-3X-3Y-B	B-5Y-5X-3X-3Y-B	B-5Y-5X-3X-3Y-B
Route2	A-4X-4Y-1Y-2Y-2X-A	A-4X-4Y-1Y-2Y-2X-A	A-4X-4Y-1Y-2Y-2X-A

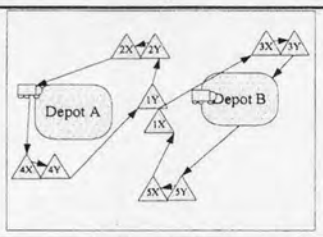
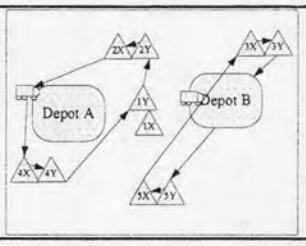
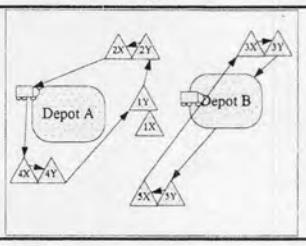
Period	1	2	3
Route			

Figure 5.12 Routes after updating

5.3 Performance measurement of heuristic

5.3.1 Data generating

5.3.1.1 Demand

Demand of each item at each outlet is deterministic. In this study, we do experiments on uniform distribution with range (15, 25) for problem E-n4-k2 and E-n5-k2. For other problem the demand is generated following standard library of Christofides and Eilon (1969).

5.3.1.2 Distance dependent cost

This cost will occur when there are arcs from a location to another location. We set cost equal to distance. The distance we calculate by the Euclidian distance between locations which derived from coordination (x, y) of location. The coordination is used follow standard library of Christofides and Eilon (1969) except E-n4-k2 and E-n5-k2. In those cases we randomly generate coordination of all locations as shown in Table 5.8-5.9 and Figure 5.13-5.14.

Table 5.8 Coordination of all depots and outlets in problem E-n4-k2

Location of		Coordination	
		X	Y
Depot	A	0	1
	B	-1	-1
Outlet	1	7	0
	2	2	9
	3	-8	0
	4	0	-8

Table 5.9 Coordination of all depots and outlets in problem E-n5-k2

Location of		Coordination	
		X	Y
Depot	A	1	1
	B	-1	-1
Outlet	1	8	0
	2	2	11
	3	-8	0
	4	0	-8
	5	-2	12

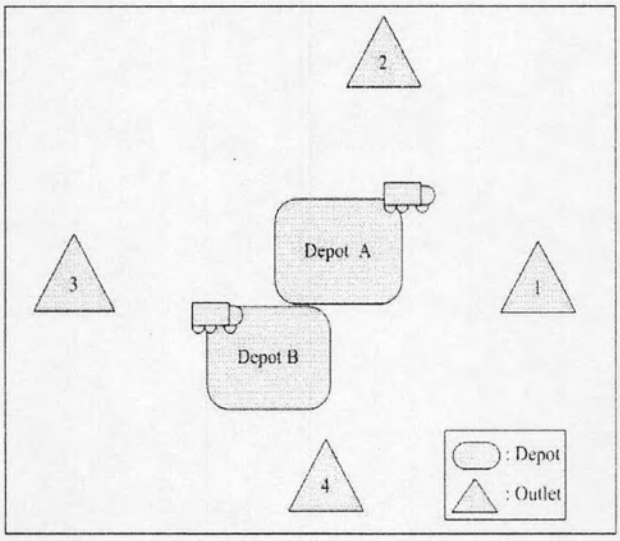


Figure 5.13 Illustration of problem E-n4-k2

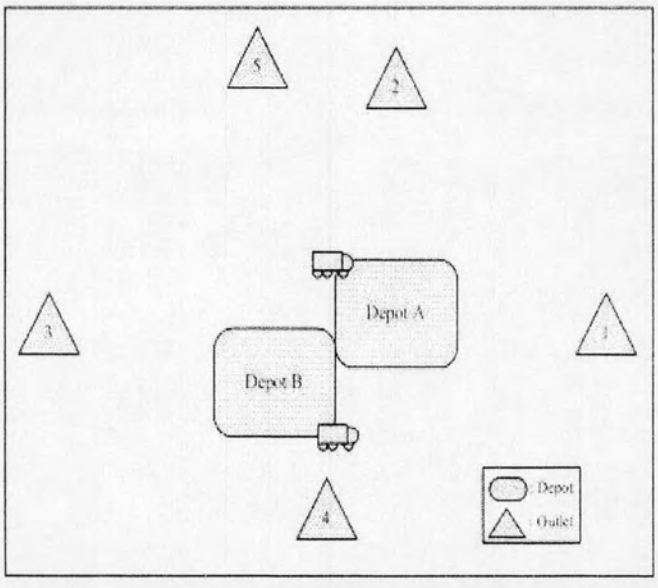


Figure 5.14 Illustration of problem E-n5-k2

The test data from standard library of Christofides and Eilon (1969), which is mentioned in the section 5.3.1.1 and 5.3.1.2, contains problem size as shown in Table 5.10.

Table 5.10 Problem size from standard library

Problem Code	Number of vehicle	Number of outlet
E-n7-k2	2	6
E-n13-k4	4	12
E-n22-k4	4	21
E-n31-k7	7	30
E-n33-k4	4	32
E-n51-k5	5	50
E-n76-k7	7	75
E-n101-k8	8	100

5.3.1.3 Vehicle fixed-charge cost

This cost will occur when there are arcs from depot to item-outlet by vehicles. We set this value equal to 30 for all problems.

5.3.1.4 Holding cost per unit per period

This cost will occur when the outlet holds inventory in the end of period. The holding cost are randomly generated according to the uniform distribution of (0.02, 0.5) for all problems.

5.3.1.5 Vehicle carrying capacity

This is the limit of vehicle to carry items in each delivery service. All vehicles have identical carrying capacity. The capacity is randomly generated according to the uniform distribution of (1.2, 10) times of ratio of maximum summation of demand in each period and number of vehicle for all problems. Ex. Assume Sum demand in period 3 which equals to 120 has the most value among all periods, if there are three vehicles in this system so the ratio of

maximum summation of demand in each period and number of vehicle is 120/3 = 40 units. The capacity is calculate by randomly selected from uniform (1.2, 10) multiply to 40. Assume that number 5 is randomly selected so capacity is 5x40=200 units per vehicle. We have to consider at maximum level of sum of demand because we avoid the case of infeasible due to the demand exceed carrying capacity in case of Lot-For-Lot solution.

5.3.1.6 Outlet storage capacity

This is the maximum inventory level that an outlet can hold amount items in each period. All outlets have different storage capacity from each other. The storage capacity is randomly generated according to the uniform distribution of (1, 5) times of maximum summation of demand of item-outlet which related to the outlet in each period. Ex. Assume an outlet "1" has item-outlet named "1X" and "1Y" related to it. Sum demand of these item-outlets in period 2 which equals to 20 has the most value among all periods. The capacity is calculate by randomly selected from uniform (1, 5) multiply to 20. Assume that number 3 is randomly selected so capacity is 3x20=60 units. We have to consider at maximum level of sum of demand because we avoid the case of infeasible due to the demand exceed storage capacity in case of Lot-For-Lot solution.

5.3.2 Performance of algorithms on small-sized test problems

The purpose of these experiments is to evaluate the performance of the proposed algorithm on the small-sized test problems that CPLEX can find solution in an acceptable time. The acceptable solving time is that the CPLEX can find at least an integer solution within 3 hours. This dissertation generates two problem sizes and modifies a test problem from library by adding some parameters that the library does not provide. All small-sized problems are listed in the table 5.11. Twenty four test instances will be generated for all problems to test the performance of heuristic.

Table 5.11 Small-sized problem

Problem Code	Number of Vehicle	Number of retailer
E-n4-k2	2	4
E-n5-k2	2	5
E-n7-k2*	2	6

* These problem use data from library of Christofides and Eilon (1969)

For the performance testing of small-sized problem, this dissertation compares solution from proposed heuristics with solution from CPLEX and from Lot-For-Lot policy (LFL). The performance of heuristics will be tested in two aspects; solution quality, and computational time. The performance of heuristics will be presented in form of the percentage deviation of a particular heuristic from the solution of CPLEX and Lot-For-Lot policy (LFL).

The heuristic have been implemented in the Visual Basic 6 programming language on a PC with an Intel Pentium 4 1.8 GHz CPU and 256 MB of RAM. The solution obtained by means of the 0-1 mixed linear integer programming formulation given in Chapter III is found by AMPL/CPLEX 8.0.0 solver. However, the CPU time is limited to at most 3 hours.

The performance of heuristic are presented by overall performance of each heuristic for solving problem in all parameter sets, the effect of parameter and interaction of parameter on performance of heuristic. The overall performances of heuristic are presented in Table 5.12-5.13. From the results, the heuristic SIOH obtains solution with 0%-0.16% average improvement from Lot-For-Lot policy and 0%-0.41% gap from solution solved by CPLEX. The negative value of gap in Table 5.13 means the proposed heuristic perform better than CPLEX. It can be considered as percentage of improvement from the solution obtained by CPLEX. For example the minimal gap from CPLEX for the problem E-n4-k2 is -0.51% means the proposed heuristic yield 0.51% better than the solution obtained by CPLEX It can be noticed from table of result that the computation time of heuristic is much better than time consumed by CPLEX.

Table 5.12 Overall Performance of SIOH for small-sized problem compared to LFL

Problem	Improvement (%) vs. Lot-for-Lot			CPU Time (Sec.)	
Name	Max	Min	Average	Lot-for-Lot	Heuristic
E-n4-k2	2.50%	0.00%	0.16%	0.01	0.07
E-n5-k2	0.00%	0.00%	0.00%	0.00	0.09
E-n7-k2	0.00%	0.00%	0.00%	0.01	0.15

Table 5.13 Overall Performance of SIOH for small-sized problem compared to CPLEX

Problem	Gap (%) vs. CPLEX			CPU Time (Sec.)	
Name	Max	Min	Average	CPLEX	Heuristic
E-n4-k2	3.04%	-0.51%	0.25%	10800	0.07
E-n5-k2	0.00%	0.00%	0.00%	10800	0.09
E-n7-k2	0.89%	0.00%	0.41%	10800	0.15

5.3.3 Performance of algorithms on medium- and large-sized test problems

The purpose of these experiments is to evaluate the performance of the proposed algorithm on the medium- and large-sized test problems that CPLEX can not find solution within 3 hours. This dissertation modifies some test problems from library by adding some parameters that the library does not provide. All medium- and large-sized problems are listed in the table 5.14 and 5.15 respectively. Twenty four test instances will be generated for all problems to test the performance of heuristic.

Table 5.14 Medium-sized problem

Problem Code	Number of Vehicle	Number of retailer
E-n13-k4*	4	12
E-n22-k4*	4	21
E-n31-k7*	7	30
E-n33-k4*	6	32

*These problems use data from library of Christofides and Eilon (1969)

Table 5.15 Large-sized problem

Problem Code	Number of Vehicle	Number of retailer
E-n51-k5*	10	50
E-n76-k7*	15	75
E-n101-k8*	20	100

*These problems use data from library of Christofides and Eilon (1969).

For the performance testing of medium- and large-sized problem, this dissertation compares solution from proposed heuristics with solution from Lot-For-Lot policy. The performance of heuristics will be tested in two aspects; solution quality, and computational time. The performance of heuristics will be presented in form of the percentage improvement of a particular heuristic from the solution of Lot-For-Lot (LFL). From the results in Table 5.16, it can be observed that the proposed algorithm can yield a good outcome for medium-sized problem. The heuristic SIOH obtains solution with 0.15%-3.78% average improvement from Lot-For-Lot policy. The results in Table 5.16 also show that the heuristics consume computation time more than Lot-For-Lot policy. However the computational time is not over than 48.79 second which is acceptable.

Table 5.16 Overall Performance of SIOH for medium-sized problem compared to LFL

Problem	Improvement (%) vs. Lot-for-Lot			CPU Time (Sec.)	
Name	Max	Min	Average	Lot-for-Lot	Heuristic
E-n13-k4	2.03%	0.00%	0.15%	0.05	0.80
E-n22-k4	2.59%	0.00%	0.20%	0.22	7.42
E-n31-k7	10.50%	0.00%	3.78%	0.77	29.73
E-n33-k4	2.77%	0.00%	0.68%	0.49	48.79

From the results in Table 5.17, the heuristic SIOH obtains solution with 3.28% average improvement from Lot-For-Lot policy for large-sized problem. The results in Table 5.17 also show that the heuristics consume much more computation time than Lot-For-Lot policy. However, the reduction on total cost is

worthy for large-sized problem in which amount of saving on total cost is much more than in small- and medium-sized problem. Hence the increase of computational time to achieve the better solution for these cases is acceptable as long as the computational time does not exceed the reasonable level.

Table 5.17 Overall Performance of SIOH for large-sized problem compared to LFL

Problem	Improvement (%) vs. Lot-for-Lot			CPU Time (Sec.)	
Name	Max	Min	Average	Lot-for-Lot	Heuristic
E-n51-k5	4.14%	0.00%	2.19%	2.35	137.08
E-n76-k7	2.93%	0.00%	0.60%	7.39	383.92
E-n101-k8	3.89%	0.00%	1.95%	18.94	1057.52

5.3.4 Discussion of computation results

It is observed from the computational results that even there are some cases that the proposed heuristic can improve the solution from Lot-For-Lot policy; the heuristic can only achieve a small improvement of Lot-For-Lot solution. This may be that the heuristic considers the replenishment quantity and delivery route of each item-outlet while most item-outlets which are in the same outlet are scheduled in the same route. When those item-outlets are served by the same route, its estimated setup cost is equivalent to zero. Since the estimated setup cost is derived from the difference between route cost of the route that includes the considered item-outlet and the route that does not include it. The cost of any route usually depends on the location of all item-outlets which are assigned to be served within it. If an item-outlet is removed from route while the other item-outlet which is in the same outlet remains in the route, the route cost will not be changed because the locations of item-outlets in the route are not changed.

Therefore, the case that the proposed heuristic can improve cost is the case that there are some item-outlets that are separated into different routes. With this structure of route removing some item-outlets can result in reduction on route cost

which can make the replenishment on these considered item-outlets in prior period to cover later period is more suitable than replenishment follow Lot-For-Lot policy.

5.4 Conclusion

This chapter has presented a heuristic for solving the multi-item multi-depot inventory routing problem. This heuristic is modified from the heuristic proposed in chapter IV. The modification of the heuristic is needed in development of solution approach. The system with outlets with multiple items are transformed into set of item-outlet so each item in each outlet is considered as an item-outlet which means it is considered as an individual outlet. This make the heuristic in chapter IV can easily be adapted to solve the multi-item multi-depot inventory routing problem.

The problem considered in this problem has the limited on both vehicle carrying capacity and outlet storage capacity. This makes the heuristic must be modified to prevent the violation of the capacity constraints. The performance of the heuristic is tested by a set of randomly generated data. The results show that the heuristic can perform well just in some cases. Only for the cases that all item-outlets which related the same outlet are separated to be served by different route. That means there is room for that item-outlet to remove from route with reduction on routing cost. In other cases, removing an item-outlet from route will take no effects on cost reduction.

Therefore, the heuristic must be developed to solve the problem with the good performance. The next chapter will present another heuristic to solve this problem. The heuristic is the modified version of this heuristic to handle the case that item-outlets which are in the same outlet are served by the same route.