CHAPTER VII

ALL ITEM-OUTLETS IN THE SAME VEHICLE HEURISTIC (AIOVH)

7.1 Introduction

Even though the results in chapter VI showed that the heuristic AIOOH can perform well, it is believed that there are some cases that the solution can be more improved by some modifications of the heuristic. As a consequence, this chapter develops another heuristic for the multi-item multi-depot inventory routing problem in order to improve the performance of the heuristic AIOOH. The new heuristic named "AIOVH" is developed by the concept of fixed-charge cost reduction. The concept will be explained by examples and illustrations during this chapter.

The remainder of this chapter is organized as follows. Opportunity to improve solution obtained by AIOOH is presented in Section 7.2. The heuristic description is presented in Section 7.3. Computational results are reported in Section 7.4. Finally, some concluding remarks are provided in Section 7.5.

7.2 Opportunity to improve solution

According to the results from chapter VI, the heuristic AIOOH outperform the heuristic H1 which is proposed in chapter V. However, there still are some opportunities to improve the solution obtained by AIOOH. It is noticed that during the process of the AIOOH, there are some cases the AIOOH can not improve the solution due to the structure of route. The route basically composes of a sequence of item-outlets and the route cost will mainly depend on the location of these item-outlets. The location of item-outlets is the location of outlet that they relate to. For example the item-outlets "1X" and "1Y" are located at the location of the outlet "1". Hence it

can be concludes that the route structure will depend mainly on the location of outlets. In the cases that the outlets in the route are located very close to another item-outlet until the process of consideration of a group of item-outlets which is used in AIOOH may cause the approximated ordering cost is very small compare to increasing holding cost. An example of this case is shown in Figure 7.1. It can be noticed that the reduction on route cost of both the group of item-outlets "1X" and "1Y" and the group of "2X" and "2Y" are very small compare to the increasing holding cost even for one period. However if all item-outlets in the route are considered together, the cost will be significantly reduced. Because the fixed-charge cost is an additional cost that will be reduced besides the route cost as shown in Figure 7.2. Hence with adding the consideration of all item-outlets in a route, the AIOOH can be modified into a new heuristic named "AIOVH".

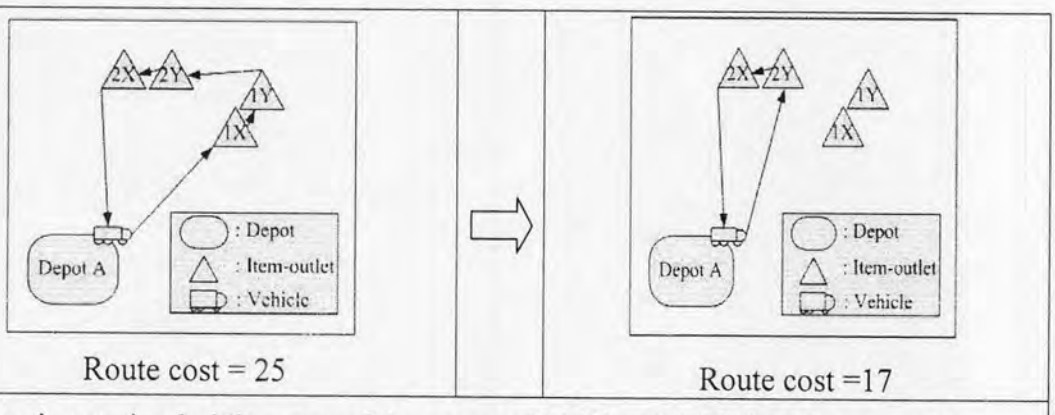
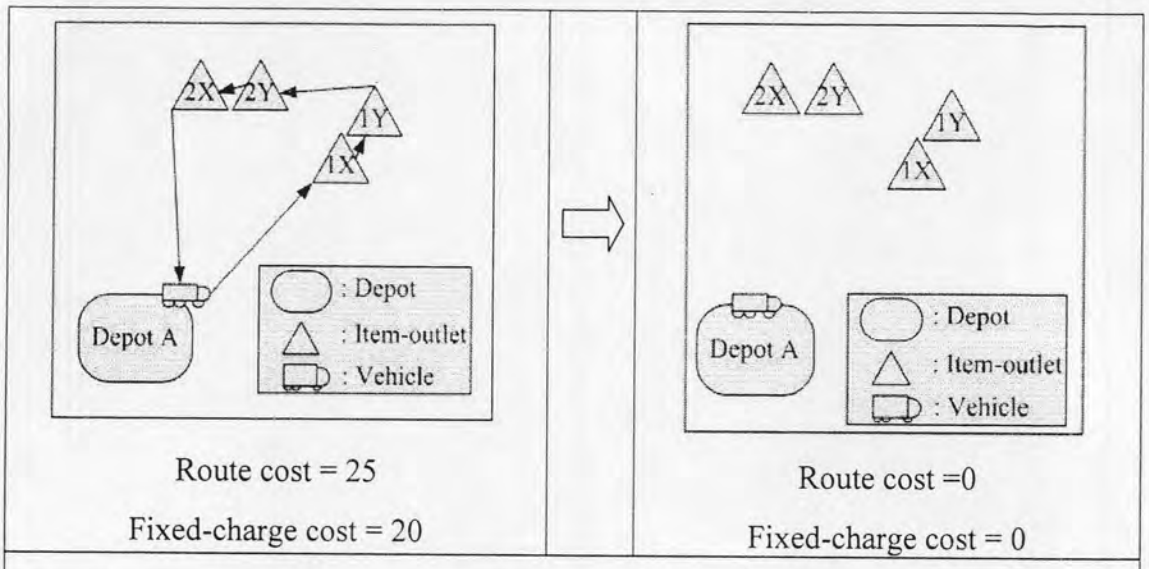


Figure 7.1 Illustration of the case that the outlets in the route are located very close to another item-outlet



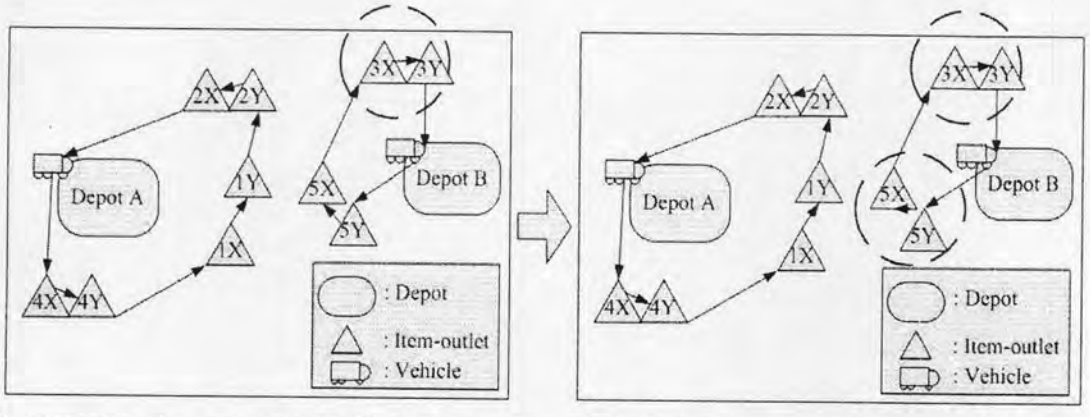
Assume increasing holding cost of the group of "1X" and "1Y" and the group of "2X" and "2Y" in this case = **10** and **15** consecutively. The total increasing cost is **10+15** = **25**. The route cost is reduced = **25** and the fixed- charge is reduced = **20**. The total reduced cost is **25+20** = **45** which is more than increasing holding cost.

Figure 7.2 Illustration of the reduction of fixed-charge cost

7.3 Development of AIOVH

7.3.1 Heuristic description

The heuristic procedure of AIOVH will start from initial solution by Lot-For-Lot as the procedure of AIOOH. The difference between the heuristic AIOVH and AIOOH is mainly in the Step 2. The Step 2 of AIOVH will randomly select a route which all item-outlets which are in the route. The illustration of difference on the selection procedure of the two heuristic is shown in Figure 7.3. Figure 7.3(a) shows the selection procedure of AIOOH that select a group of item-outlets "3X" and "3Y" which is in the same route and in the same outlet named "3" while Figure 7.3(b) shows that AIOVH select all item-outlets in a route which composes of "3X", "3Y", "5X" and "5Y". In the later procedure of AIOVH, the replenishment quantity of this group of item-outlets will be determined by solving the Lot-sizing problem of this group.



(a) AIOOH selects a group of item-outlets.

(b) AIOVH selects all item-outlets in a route.

Figure 7.3 Difference between AIOOH and AIOVH in item-outlet selection process

With the consideration of all item-outlets in a route there are two cases that the total transportation cost which is the summation of route cost and fixed-charge cost can be reduced. The first case that there is reduction on transportation cost is that the remaining carrying capacity of at least one vehicle is enough for all summation of demand of a set of item-outlets which are in the same route in later period. It is sure that if they are removed from route in later period they will cause the reduction on routing and fixed-charge cost on that period. The remaining capacity of vehicle must be enough to take this amount of demand of this set of item-outlets. This case the demand that are removed in later period and add in prior period can be taken by remaining capacity of vehicles with out re routing. With this case the routing cost of prior period will be the same so the reduction of transportation will come from fixed-charge cost that is reduce from less use of vehicle in late period and routing cost in that vehicle. The illustration of this case will be provided in Figure 7.4. Example of this case is the case that there is a vehicle serving a few number of item-outlet and summation of load in that vehicle is small compare to remaining capacity in vehicle in prior period. So all item-outlets can be move to prior period to send replenishment quantity to cover demand with out additional vehicle to use.

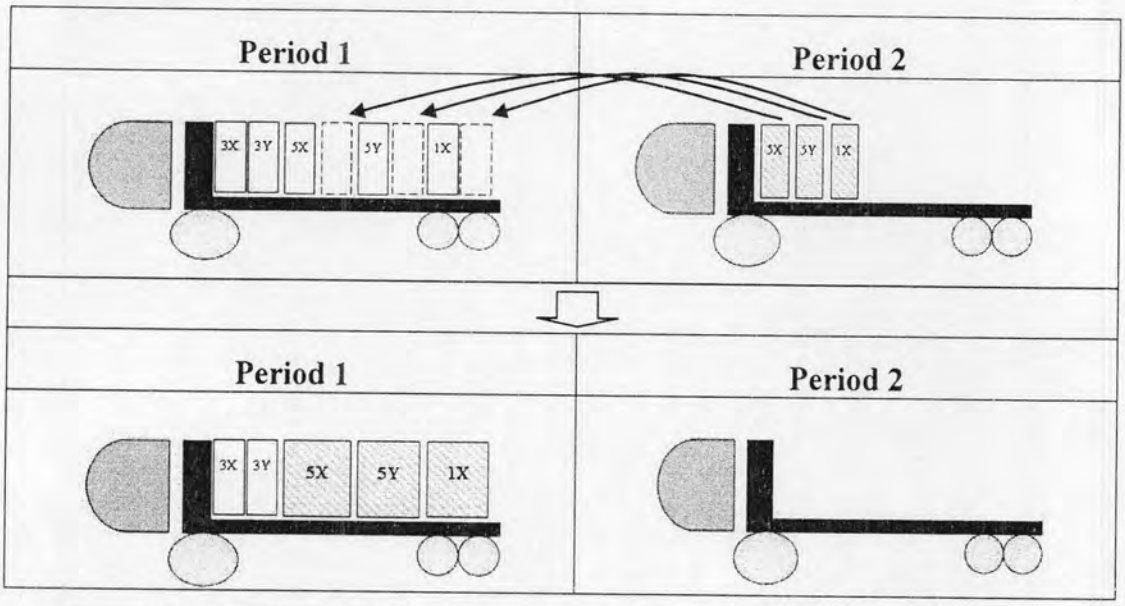


Figure 7.4 Illustration of the case that remaining capacity is enough for demand of all item-outlets in a vehicle

The last case is that item-outlets in the considered route can not be added in the same route in the prior period but can be separately added into many routes. In this case the remaining capacity of one vehicle is not enough for total demand of all item-outlets in the considered route but each existing vehicle has enough remaining capacity to be filled by demand of some item-outlets. In this case the route of prior period may be changed and result in additional route cost but it is less than the reduced fixed-charge cost. For example assume that load in a vehicle in period 2 is 30 units and there is a vehicle in period 1 with remaining capacity 25 units and other vehicles with 10 units remain capacity. In this case the move of all loads in the vehicle in period 2 to period 1 may cause the route in period 1 changed. However the cost that can be reduced from fixed-charge and routing cost is more that the additional cost of rerouting in period 1 so this move is worthy to be considered. The illustration of this case is shown in Figure 7.5.

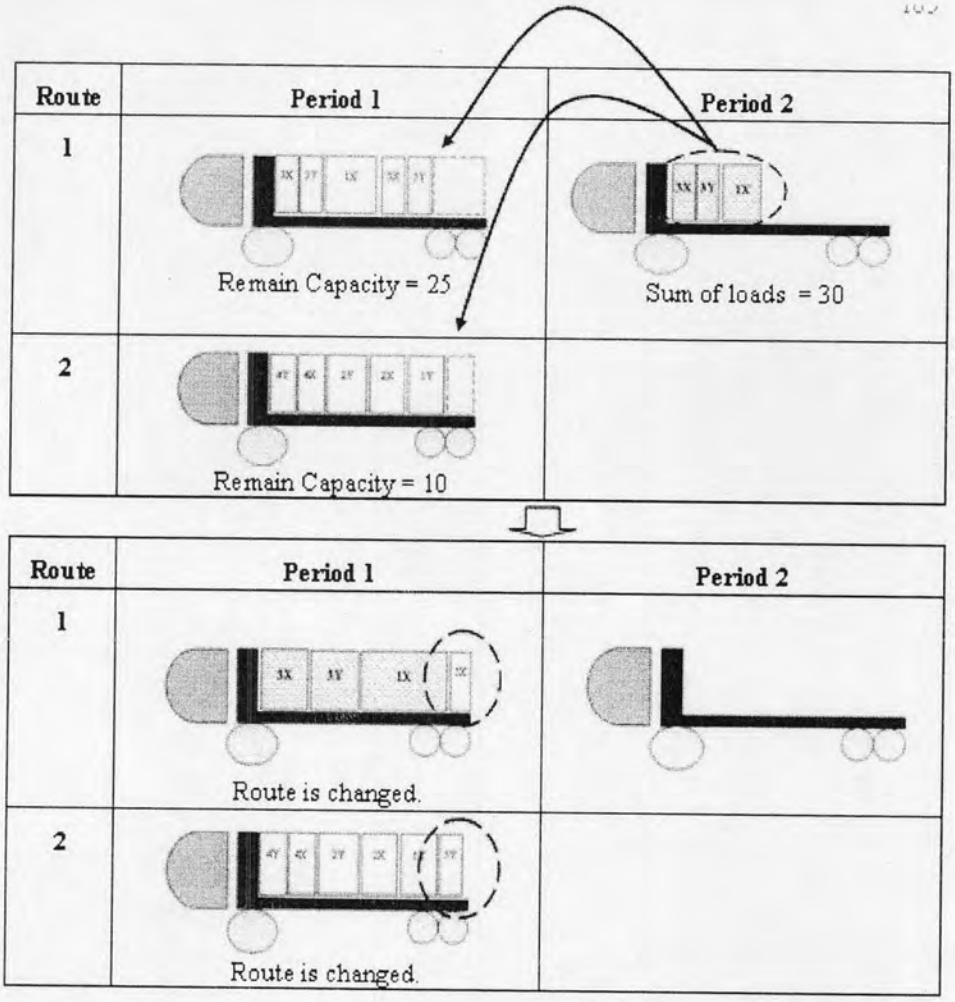


Figure 7.5 Illustration of all loads moved with rerouting in prior period

7.3.2 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 7.6. Figure 7.6 illustrates the graphical location of all depots and item-outlets in the example problem. Table 7.1 shows the distance between all locations. Demand of all outlets is given in Table 7.2. Table 7.3 gives the information about the holding cost of the outlets. Beginning inventory of all outlets are assumed to be zero.

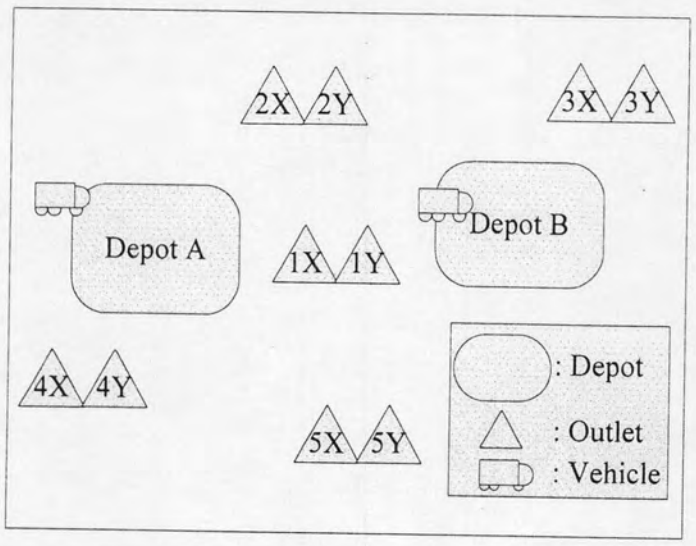


Figure 7.6 Illustration of the graphical location of all depots and outlets in the example problem.

Table 7.1 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 7.2 Demand of all outlets

		Period	**1**	**2**	**3**
		Item			
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 7.3 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 7,4 and Figure 7.7. Table 7.4 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 7.4 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

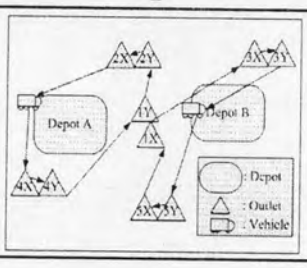
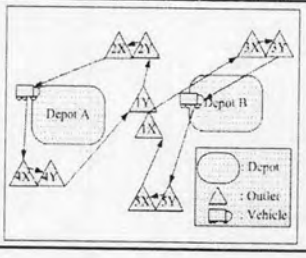
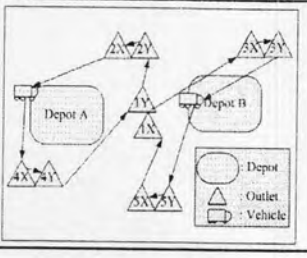
Period	1	2	3
Route			

Figure 7.7 Initial Route

Step 2: select an outlet

The illustration of selection of an item-outlet is shown in Figure 7.8. In Figure 7.8, a set of all item-outlet in the route 2 which contains "4X", "4Y", "1Y", "2Y" and "2X" is selected to be considered for determining replenishment quantity of them. The shortest path network is used to determine the replenishment quantity.

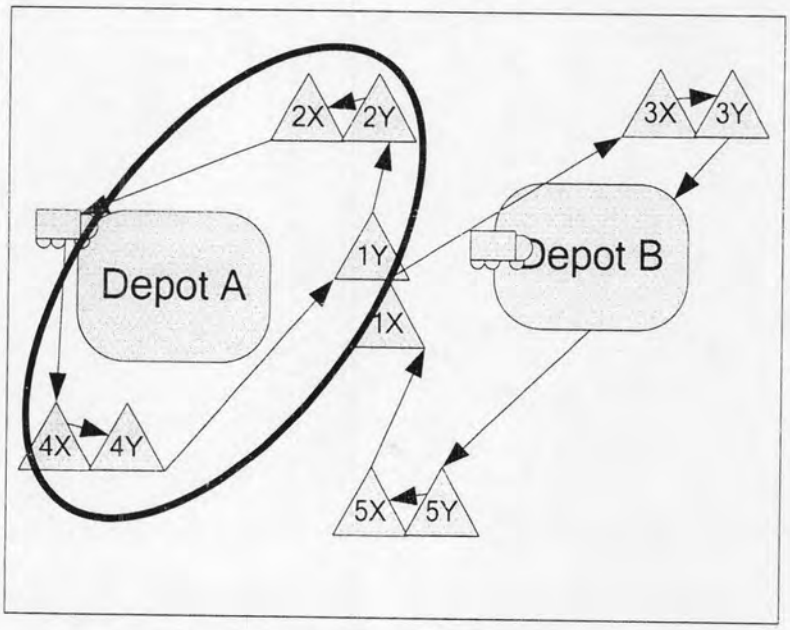


Figure 7.8 A set of item-outlet is selected

Step 3: Determine delivery period using shortest path algorithm.

Step 3.1 Define Node and Arc.

The illustration of shortest path network is presented in the Figure 7.9.

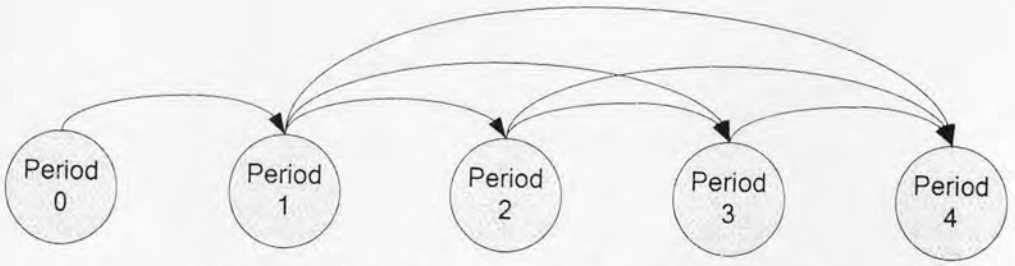


Figure 7.9 Illustration of shortest path network

For Figure 7.9, each circle 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 set of all item-outlet in the route 2" is replenished at period 3 after replenishment at period 2. For the case of arc 0-t means the set of all item-outlet in the route 2 is first replenished at period t ex. an arc 0-1 means the first replenishment to the set of all item-outlet in the route 2 is made at period 1. For the case of arc k-4, which node 4 is the last node in this problem, means the set of all item-outlet in the route 2 is last replenished at period k ex. an arc 2-4 means the last replenishment to the set of all item-outlet in the route 2 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 set of all item-outlet in the route 2 is shown in Figure 7.11.

Table 7.5 Holding Cost of Replenishment Period following Arc

To Period / From Period	1	2	3	4
0	0	-	-	-
1	-	0	8x0.35+10x0.4+9x0.15 +10x0.3+8x0.25=13.15	(17+9)x0.35+(21+11)x0.4 +(19+10)x0.15+(21+11)x0.3 +(17+9)x0.25=42.35
2	-	-	0	9x0.35+11x0.4+10x0.15 +11x0.3+9x0.25=14.6
3	-	-	-	0

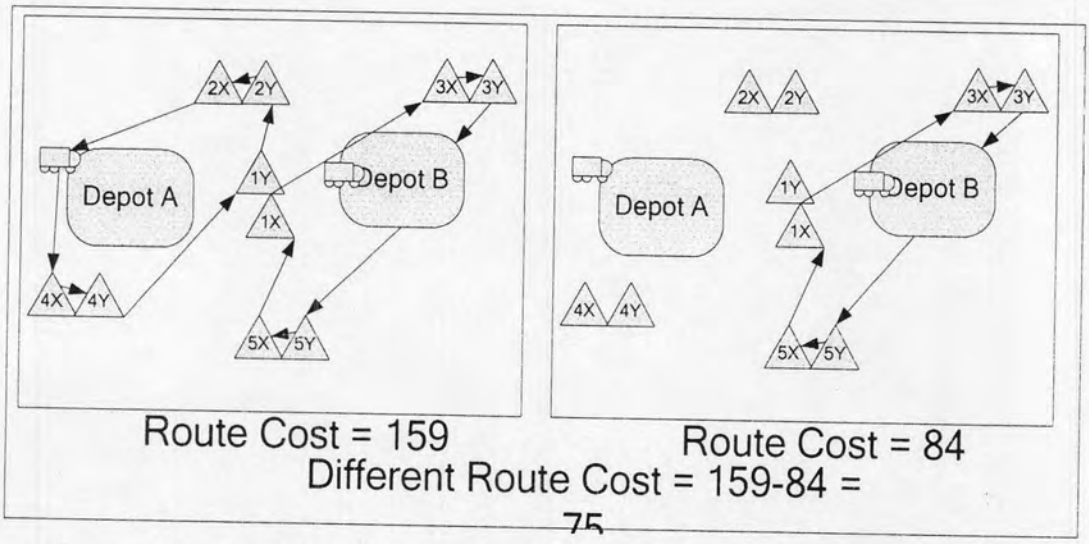


Figure 7.10 Illustration of different route cost in period 2

Table 7.6 Arc Cost of set of all item-outlet in the route 2

To Period / From Period	1	2	3	4
0	0	-	-	-
1	-	0+75=75	13.15+75=88.15	42.35+75=117.35
2	-	-	0+75=75	14.6+75=89.6
3	-	-	-	0+75=75

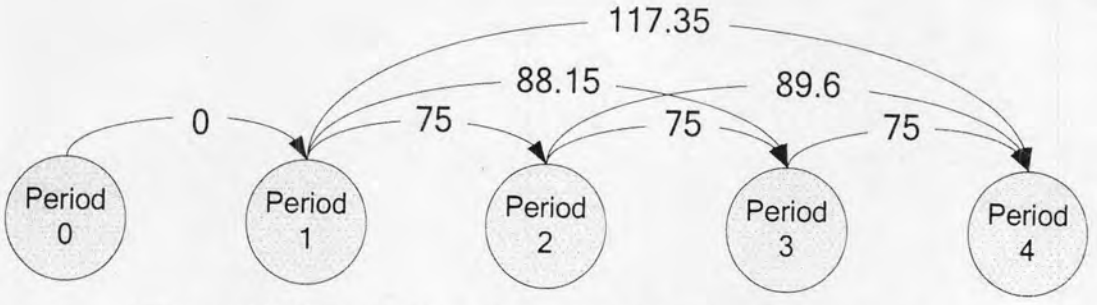


Figure 7.11 Illustration of arc cost of shortest path network

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

The solution of shortest path problem of the set of all item-outlet in the route 2 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 7.7 and Figure 7.12. In Table 7.7, the replenishment quantity is presented. It can be indicated that the replenishment quantity of the set of all item-outlet in the route 2 which contains "4X", "4Y", "1Y", "2Y" and "2X" in period 1, 2 and 3 are changed. The value at period 1 of "4X", "4Y", "1Y", "2Y" and "2X" are 27, 29, 30, 29 and 27 respectively, 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 route in period 2 and 3 remain one route B-5Y-5X-1X-3X-3Y-B that means the vehicle starts from depot indexed by "B" and goes to item-outlet "5Y", "5X", "1X", "3X", "3Y" and return to "B".

Table 7.7 Replenishment quantity and delivery route after updating

Item-Outlet \ Period	1	2	3
1X	12	8	10
1Y	11+9+10=30	0	0
2X	10+8+9=27	0	0
2Y	8+10+11=29	0	0
3X	12	8	10
3Y	11	9	10
4X	10+8+9=27	0	0
4Y	8+10+11=29	0	0
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		

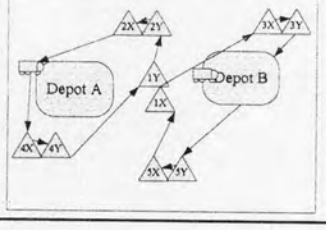
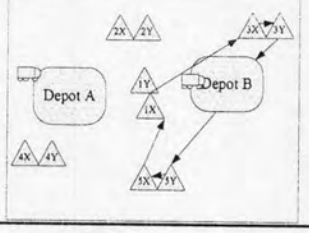
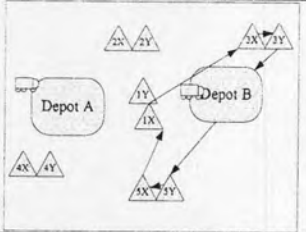
Period	1	2	3
Route			

Figure 7.12 Route after updating

7.4 Experiment and result

The performance of the heuristic AIOVH is tested by the problem instances described in chapter V and compare to the solution obtained by Lot-For-Lot policy.

7.4.1 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

within 3 hours. The test instances which are described in chapter V are used for performance evaluation of AIOVH. 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 7.8-7.11. From the results, the heuristic AIOVH obtains solution with 15.07%-26.06% average improvement from Lot-For-Lot policy and 1.61%-15.19% from solution solved by CPLEX. It can be noticed from table of result that the computation time of heuristic is much better than time consumed by CPLEX. The results in Table 7.10 indicate that the AIOVH can perform better than SIOH between 15.94% and 22.14% in average while AIOVH consumes slightly more time than SIOH. The results in Table 7.11 indicate that the AIOVH can perform better than AIOOH between 6.33% and 8.26% in average while AIOVH consumes slightly more time than AIOOH.

Table 7.8 Overall Performance of AIOVH 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	25.44%	2.50%	15.07%	0.01	0.09
E-n5-k2	38.54%	5.00%	26.06%	0.00	0.13
E-n7-k2	35.18%	6.90%	23.18%	0.01	0.23

Table 7.9 Overall Performance of AIOVH for small-sized problem compared to CPLEX

Problem	Improvement (%) vs. CPLEX			CPU Time (Sec.)	
Name	Max	Min	Average	CPLEX	Heuristic
E-n4-k2	17.48%	1.07%	5.67%	10800	0.09
E-n5-k2	5.94%	0.00%	1.61%	10800	0.13
E-n7-k2	25.65%	1.72%	15.19%	10800	0.23

Table 7.10 Overall Performance of AIOVH for small-sized problem compared to SIOH

Problem	Difference (%) on Total Cost			CPU Time (Sec.)	
Name	Max	Min	Average	SIOH	AIOOH
E-n4-k2	29.44%	2.50%	15.94%	0.07	0.09
E-n5-k2	38.54%	5.05%	21.72%	0.09	0.13
E-n7-k2	35.18%	6.90%	22.14%	0.15	0.23

Table 7.11 Overall Performance of AIOVH for small-sized problem compared to AIOOH

Problem	Difference (%) on Total Cost			CPU Time (Sec.)	
Name	Max	Min	Average	AIOOH	AIOVH
E-n4-k2	15.74%	0.00%	6.33%	0.07	0.09
E-n5-k2	15.74%	0.00%	7.62%	0.09	0.13
E-n7-k2	28.97%	0.00%	8.26%	0.18	0.23

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

The purpose of these experiments is to evaluate the performance of the AIOVH on the medium- and large-sized test problems that CPLEX can not find solution within 3 hours. The test instances which are described in chapter V are used for performance evaluation of AIOVH.

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 7.12, it can be observed that the proposed algorithm can yield a good outcome for medium-sized problem. The heuristic AIOVH obtains solution with 12.27%-24.83% average improvement from Lot-For-Lot policy. The results in Table 7.13 show that the heuristics consume computation time more than Lot-For-Lot policy. However the computational time is not over than 24.51 second which is acceptable. The results in Table 7.13 indicate that the AIOVH can perform better than SIOH between 7.88% and 18.13% in average while AIOVH consumes a few less time than SIOH. The results in Table 7.14 indicate that the AIOVH can perform better than AIOOH between 3.26% and 8.23% in average while AIOVH consumes a few more time than AIOOH.

Table 7.12 Overall Performance of AIOVH 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	38.09%	5.69%	24.83%	0.05	0.57
E-n22-k4	42.95%	1.69%	19.75%	0.22	5.50
E-n31-k7	33.02%	2.42%	12.27%	0.77	18.09
E-n33-k4	43.80%	3.29%	21.37%	0.49	24.51

Table 7.13 Overall Performance of AIOVH for medium-sized problem compared to SIOH

Problem	Difference (%) on Total Cost			CPU Time (Sec.)	
Name	Max	Min	Average	SIOH	AIOOH
E-n13-k4	38.09%	5.69%	24.31%	0.80	0.83
E-n22-k4	42.24%	0.80%	18.69%	7.42	7.79
E-n31-k7	24.19%	0.54%	7.88%	29.73	20.50
E-n33-k4	43.26%	1.72%	18.13%	48.79	27.61

Table 7.14 Overall Performance of AIOVH for medium-sized problem compared to AIOOH

Problem	Difference (%) on Total Cost			CPU Time (Sec.)	
Name	Max	Min	Average	AIOOH	AIOVH
E-n13-k4	14.45%	0.00%	4.16%	0.57	0.83
E-n22-k4	11.00%	0.00%	4.18%	5.50	7.79
E-n31-k7	12.20%	0.08%	3.26%	18.09	20.50
E-n33-k4	24.34%	0.00%	8.23%	24.51	27.61

From the results in Table 7.15, the heuristic AIOVH obtains solution with 15.43%-22.8% average improvement from Lot-For-Lot policy. The results in Table 7.15 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. The results in Table 7.16 indicate that the AIOVH can perform better than SIOH between 15.2% and 17.71% in average while AIOVH consumes much less time than SIOH. The results in Table 7.17 indicate that the AIOVH can perform better than AIOOH between 2.09% and 9.73% in average while AIOVH consumes a few more time than AIOOH.

Table 7.15 Overall Performance of AIOVH 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	33.00%	3.73%	17.21%	2.35	89.98
E-n76-k7	29.54%	2.78%	15.43%	7.48	260.61
E-n101-k8	37.39%	5.90%	22.80%	18.94	653.99

Table 7.16 Overall Performance of AIOVH for large-sized problem compared to SIOH

Problem	Difference (%) on Total Cost			CPU Time (Sec.)	
Name	Max	Min	Average	SIOH	AIOOH
E-n51-k5	30.87%	1.96%	15.20%	137.08	89.98
E-n76-k7	29.15%	1.97%	15.41%	390.43	260.61
E-n101-k8	35.19%	2.82%	17.71%	1057.52	653.99

Table 7.17 Overall Performance of AIOVH for large-sized problem compared to AIOOH

Problem	Difference (%) on Total Cost			CPU Time (Sec.)	
Name	Max	Min	Average	AIOOH	AIOVH
E-n51-k5	24.20%	0.00%	5.36%	85.08	89.98
E-n76-k7	5.93%	0.24%	2.09%	217.17	260.61
E-n101-k8	34.82%	0.56%	9.73%	604.65	653.99

7.5 Conclusion

This chapter indicates the opportunity to improve the heuristic AIOOH where the cases that the route cost and fix-charged cost can be reduced in presented. After that the heuristic AIOVH which is the improved version of AIOOH is presented. The computational results of AIOVH are provided and discussed in later section. The results show that the heuristic AIOVH can perform better than AIOOH in many cases with moderately more computational time. It can be concluded that the heuristic AIOVH is the better alternative of AIOOH. It is due to the modification in item-outlet selection. Besides selection a group of item-outlet which is in the same route and in the same outlet, AIOVH considers all item-outlets which are in the same route. This difference of the two heuristics results in different performance of them. The next chapter will present the conclusion of this dissertation.