Chapter 4

4 COMPUTATIONAL RESULTS

The VRP problem is NP-hard and making use of heuristic methods results in unpredictable results. Heuristic methods are non deterministic which contribute to the complexity in measuring the effectiveness of the method applied on the problem.

The VRP with additional side constraints is a complex problem that complicated basic rules specified for the guidance algorithm of the applied meta-heuristic. Depending on the distribution of data points, time windows, peak and off-peak travel times, vehicle capacity and demand per stop, the algorithm must adapt to the data environment during the execution to result in an acceptable feasible solution. To achieve this, we implemented a multiple operation selection method. We projected that there must be an effective operation in our list of operations on the data environment. In the previous chapter, we discussed the methods and proof theoretically that the proposed solution will be effective. In this chapter we will discuss the impact of the operations on the problem, as well as the additional advantage obtained by using these operations in combinations.

The implementation of the algorithm consist of two phases: the initial solution make use of the Sequential Insertion Heuristic to construct a set of initial routes and the improvement heuristic consist of a hybrid method based mainly on the Tabu Search technique and the Simulated Annealing method. Although we are interested in the improvement heuristic, we will present the results of the construction heuristic to indicate the efficiency of the improvement heuristic.

Results will be presented for two types of problems:

1. The traditional Solomon benchmark problems will be solved to indicate the efficiency of the algorithm with known results.

2. A real-life problem will be solved and efficiency will be discussed.

The chapter will discuss the results of the initial solution, the effect of the individual operations of the improvement heuristic and the results of the improvement phase.

4.1. Solomon's Benchmark Problems

Solomon generated six sets of problems. Their design highlights several factors that affect the behaviour of routing and scheduling algorithms. They are: geographical data; the number of customers serviced by a vehicle; percent of time-constrained customers; and tightness and positioning of the time windows.

The geographical data are randomly generated in problem sets R1 and R2, clustered in problem sets C1 and C2, and a mix of random and clustered structures in problem sets by RC1 and RC2. Problem sets R1, C1 and RC1 have a short scheduling horizon and allow only a few customers per route (approximately 5 to 10). In contrast, the sets R2, C2 and RC2 have a long scheduling horizon permitting many customers (more than 30) to be serviced by the same vehicle.

The customer coordinates are identical for all problems within one type (i.e., R, C and RC). The problems differ with respect to the width of the time windows. Some have very tight time windows, while others have time windows, which are hardly constraining. In terms of time window density, that is, the percentage of customers with time windows, he created problems with 25, 50, 75 and 100 % time windows.

The larger problems are 100 customer euclidean problems where travel times equal the corresponding distances. For each such problem, smaller problems have been created by considering only the first 25 or 50 customers. We only consider the larger problems.

4.1.1. *Initial solution.*

High quality initial heuristics often allow local searches and metaheuristics to achieve better solutions more quickly. We implemented the sequential insertion heuristic (SIH) proposed by Marius Solomon. We extended the Solomon criteria by utilising the neighbours stop information in testing for a suitable stop to add to the route. We also extended the criteria by a push backward if a customer is inserted between the depot and the first customer as proposed by Dullaert and Bräysy (2003).

When we start a route, the selection of the first node can be done according to the following criteria:

- Selecting the node that has the latest departure time.
- Selecting the node that has the earliest arrival time.
- Selecting the node that is the furthest from the depot.
- Selecting the node that is the closes to the depot.

The seed node criteria results in a different solution set according to the seed selection. The selection of a seed vehicle can also result in a different solution and we select the vehicle according to the following criteria:

- The vehicle with the smallest capacity.
- The vehicle with the least running cost.

Combining these two criteria, we result in eight possible initial solution generation methods. Although the implementation of all eight methods contributes to additional computation time, we can motivate the decision by the following:

- The input data is unpredictable and we cannot beforehand decide which method will be the best for the input data.
- The better the initial solution, the quicker the improvement phase. The time spend on the additional seed criteria will be made up in the improvement phase.
- The use of a neighbour list and the greedy nature of the sequential insertion heuristic result in a fix time for the initial solution.

The following table shows the initial results for the 56 Solomon benchmark problems according to the seed criteria. Because Solomon uses homogeneous fleet, only the stop criteria are considered. The highlighted text shows the best result achieved.

Problem Class C								
Problem	Latest Departure		Earliest Arrival		Furthest		Closest	
C101	10	923.70	10	880.47	10	880.47	10	928.22
C102	11	1193.46	10	997.74	11	1151.06	10	1075.08
C103	11	1317.31	11	1536.23	12	1501.95	10	1081.56
C104	10	1135.85	10	1419.31	11	1098.36	10	1059.59
C105	10	878.78	10	934.36	10	934.36	10	932.38
C106	11	1073.75	10	1068.90	10	1076.65	10	968.58
C107	10	928.74	10	1066.52	10	1066.52	10	1017.70
C108	10	871.57	10	1122.68	10	1114.62	10	1121.52
C109	10	910.28	10	1152.90	10	1285.48	11	1188.59
Problem	Latest Departure		Earliest Arrival		Furthest		Closest	
C201	3	895.38	3	1023.26	3	826.15	3	838.65
C202	3	1180.34	3	1727.11	3	1778.32	3	1774.13
C203	3	1173.25	3	1572.67	3	2091.21	3	1965.51
C204	3	1235.70	3	1498.34	3	1524.62	3	1509.96
C205	3	789.79	3	1318.07	3	1026.28	3	1170.94
C206	3	934.87	3	1456.09	3	1413.57	3	1349.79
C207	3	884.44	3	1040.77	3	1082.25	3	1140.17
C208	3	815.97	3	1201.14	3	1108.66	3	1205.91

Table 2: Solomon Initial Solution
Results Class C

Problem Class R								
Problem	Latest Departure		Earliest Arrival		Furthest		Closest	
R101	20	1857.93	23	2303.99	25	2293.20	24	2301.59
R102	19	1792.59	20	2095.62	20	1913.25	21	1956.94
R103	15	1553.58	17	1777.33	19	1777.82	17	1694.45
R104	12	1283.22	13	1516.91	12	1334.73	12	1358.23
R105	15	1534.40	16	1804.79	16	1802.51	16	1883.03
R106	15	1457.51	17	1776.22	16	1714.83	15	1715.28
R107	13	1336.79	14	1591.55	13	1488.27	14	1549.74
R108	10	1174.06	11	1284.84	12	1385.20	11	1237.10
R109	14	1423.01	14	1645.27	15	1641.27	14	1696.22
R110	12	1332.66	14	1620.57	15	1682.08	13	1577.98
R111	13	1344.17	15	1672.66	15	1652.95	13	1606.67
R112	11	1167.79	12	1475.42	12	1436.07	11	1335.74
Problem	Latest Departure		Earliest Arrival		Furthest		Closest	
R201	4	1791.78	5	1633.98	5	1822.54	5	2043.84
R202	4	1603.75	5	1703.38	5	1623.01	5	1570.04
R203	4	1325.15	4	1505.26	4	1602.95	4	1518.06
R204	3	1054.39	3	1146.17	3	1183.05	3	1107.48
R205	4	1551.95	4	1461.61	4	1467.55	4	1533.07
R206	3	1358.68	3	1364.04	4	1501.83	3	1378.20
R207	3	1205.44	3	1213.78	3	1272.68	3	1279.97
R208	3	954.38	3	985.00	3	908.49	3	945.51
R209	4	1441.55	4	1409.81	4	1339.33	4	1260.75
R210	4	1384.77	4	1548.22	4	1510.95	4	1478.19
R211	3	1080.89	3	1200.61	3	1173.58	3	1213.90

Table 3: Solomon Initial Solution Results Class R

Problem Class RC								
Problem	Latest Departure		Earliest Arrival		Furthest		Closest	
RC101	16	1929.02	17	2186.36	18	2065.91	16	2050.29
RC102	15	1789.29	16	2134.40	17	1900.65	17	2235.96
RC103	13	1613.99	14	1924.30	15	1765.69	15	1960.02
RC104	12	1363.74	13	1731.69	13	1524.04	13	1677.67
RC105	16	1805.33	18	2299.15	19	2236.09	17	2146.35
RC106	14	1581.39	15	1940.90	14	1932.27	16	2135.60
RC107	13	1607.96	14	1881.29	15	1896.47	14	1823.81
RC108	12	1340.10	13	1728.38	13	1626.48	13	1639.01
Problem	Latest Departure		Earliest Arrival		Furthest		Closest	
RC201	5	2213.00	5	2273.59	5	2272.32	5	2131.14
RC202	5	1943.42	5	2203.85	5	1953.77	5	2031.00
RC203	4	1727.98	4	1595.85	4	1692.00	4	1758.04
RC204	3	1217.82	4	1449.52	4	1464.28	3	1184.48
RC205	6	1940.44	5	2137.55	5	2396.53	5	2151.34
RC206	4	1691.69	4	1723.34	4	1631.19	4	1595.74
RC207	4	1731.50	4	1690.61	4	1491.13	4	1627.06
RC208	3	1275.21	3	1347.44	3	1347.62	3	1564.00

Table 4: Solomon Initial Solution
Results Class RC

4.1.2. Improvement Phase

The previous paragraph has shown the effectiveness of the individual operators. The purpose of the improvement phase is to combine these individual operators such that we can achieve effective improvements. The utilisation of the operators in random combination with each other result in a robust method that achieve results faster.

The following table shows the results compared to the best-published Solomon results as well as the initial result the improvement heuristic started from.

Problem Class C									
Problem	Initial Solution		Improvement			Best Published			
C101	10	880.47	10	828.94	5.9%	10	828.94	0.0%
C102	10	997.74	10	871.32	12.7%	10	828.94	5.1%
C103	10	1081.56	10	916.83	15.2%	10	828.06	10.7%
C104	10	1059.59	10	911.85	13.9%	10	824.78	10.6%
C105	10	878.78	10	827.55	5.8%	10	828.94	-0.2%
C106	10	968.58	10	840.19	13.3%	10	828.94	1.4%
C107	10	928.74	10	827.55	10.9%	10	828.94	-0.2%
C108	10	871.57	10	827.55	5.1%	10	828.94	-0.2%
C109	10	910.28	10	829.74	8.8%	10	828.94	0.1%
Problem	Initial Solution		Improvement			Best Published		
C201	3	826.15	3	588.88	28.7%	3	591.56	-0.5%
C202	3	1180.34	3	623.46	47.2%	3	591.56	5.4%
C203	3	1173.25	3	625.46	46.7%	3	591.17	5.8%
C204	3	1235.70	3	685.10	44.6%	3	590.6	16.0%
C205	3	789.79	3	617.45	21.8%	3	588.88	4.9%
C206	3	934.87	3	629.63	32.7%	3	588.49	7.0%
C207	3	884.44	3	587.89	33.5%	3	588.29	-0.1%
C208	3	815.97	3	592.93	27.3%	3	588.32	0.8%

Table 5: Class C Solomon Solution[4]

[4] Source: Solomon M. [45]

Problem Class R								
Problem	Initial Solution		Improvement			Best Published		
R101	20	1857.93	20	1670.13	10.1%	19	1645.79	1.5%
R102	19	1792.59	19	1576.81	12.0%	17	1486.12	6.1%
R103	15	1553.58	15	1316.31	15.3%	13	1292.68	1.8%
R104	12	1283.22	11	1061.90	17.2%	9	1007.24	5.4%
R105	15	1534.40	15	1455.08	5.2%	14	1377.11	5.7%
R106	15	1457.51	14	1292.28	11.3%	12	1251.98	3.2%
R107	13	1336.79	12	1174.00	12.2%	10	1104.66	6.3%
R108	10	1174.06	9	1030.87	12.2%	9	960.88	7.3%
R109	14	1423.01	13	1284.32	9.7%	11	1194.73	7.5%
R110	12	1332.66	13	1205.48	9.5%	10	1118.59	7.8%
R111	13	1344.17	13	1239.26	7.8%	10	1096.72	13.0%
R112	11	1167.79	11	1059.78	9.2%	9	982.14	7.9%
Problem	Initial Solution		Improvement			Best Published		
R201	5	1633.98	4	1335.55	18.3%	4	1252.37	6.6%
R202	5	1570.04	4	1200.26	23.6%	3	1191.7	0.7%
R203	4	1325.15	3	972.59	26.6%	3	939.54	3.5%
R204	3	1054.39	3	842.54	20.1%	2	825.52	2.1%
R205	4	1461.61	3	1133.02	22.5%	3	994.42	13.9%
R206	3	1358.68	3	985.94	27.4%	3	906.14	8.8%
R207	3	1205.44	3	948.50	21.3%	2	893.33	6.2%
R208	3	908.49	2	845.94	6.9%	2	726.75	16.4%
R209	4	1260.75	4	930.43	26.2%	3	909.16	2.3%
R210	4	1384.77	3	1019.45	26.4%	3	939.34	8.5%
R211	3	1080.89	3	862.42	20.2%	2	892.71	-3.4%

Table 6: Class R Solomon Solution

Problem Class RC							
Problem	Initial Solution		Improvement			Best Published	
RC101	16	1929.02	16	1742.62	9.7%	14	1696.94	2.7%
RC102	15	1789.29	15	1625.30	9.2%	12	1554.75	4.5%
RC103	13	1613.99	13	1403.99	13.0%	11	1261.67	11.3%
RC104	12	1363.74	12	1212.92	11.1%	10	1135.48	6.8%
RC105	16	1805.33	16	1706.53	5.5%	13	1629.44	4.7%
RC106	14	1581.39	14	1502.00	5.0%	11	1424.73	5.4%
RC107	13	1607.96	12	1318.22	18.0%	11	1230.48	7.1%
RC108	12	1340.10	12	1240.27	7.4%	10	1139.82	8.8%
Problem	Initial Solution		Improvement			Best Published		
RC201	5	2131.14	4	1474.86	30.8%	4	1406.91	4.8%
RC202	5	1943.42	4	1298.28	33.2%	3	1367.09	-5.0%
RC203	4	1595.85	3	1081.34	32.2%	3	1049.62	3.0%
RC204	3	1184.48	3	883.53	25.4%	3	798.41	10.7%
RC205	6	1940.44	5	1311.93	32.4%	4	1297.19	1.1%
RC206	4	1595.74	4	1162.03	27.2%	3	1146.32	1.4%
RC207	4	1491.13	4	1106.24	25.8%	3	1061.14	4.2%
RC208	3	1275.21	3	920.17	27.8%	3	828.14	11.1%

Table 7: Class RC Solomon Solution

Figure 26 displays the results in graphical format. The results are within reasonable margin from the best-published results. We must take into account that the best-published methods were achieved by various methods, i.e. for a specific problem instance, a specifically designed algorithm were applied on the problem. The comparison confirms the ability of our algorithm to perform reasonable across different problem instances.

In some instances our algorithm improved on the best-published result. From problem RC202 we can see a 5% improvement on the best published. It must be noted that the cost function was set only on distance for these instances, which could resulted in higher total cost. We can see that from the difference in number of vehicles in problems R211 and RC202.

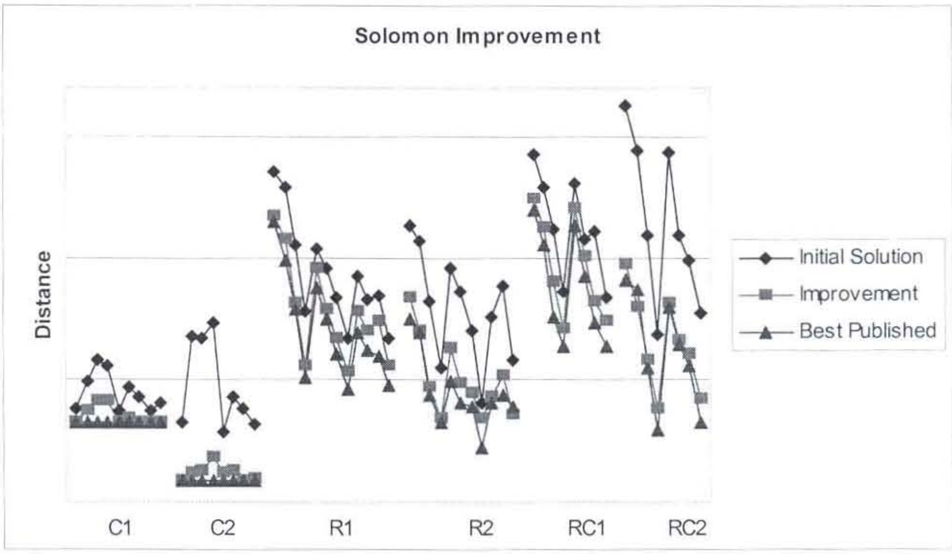

Figure 26: Solomon Improvement Comparison

4.2. Operation Results

Our algorithm was designed for the specific purpose of implementing it in the ASP environment. This environment is unpredictable in terms of input data, as well as cost factors. The idea of controlling specific operations through a meta heuristic had to be supported by a set of effective operations. Driven by the Tabu methodology, we were looking for operations that can assist as in both intensification and diversification. For this purpose we utilised some of the existing operations and designed new operations for the specific environment.

To ensure integrity of the system we tested each operation on its own to ensure that the operation acts according to expectation as well as resulting in useful neighbourhood solutions.

4.2.1. Insert Operator

This operation was added to ensure that we have viable routes by adding all the orphans available on the existing routes, or by creating new routes if the first is not viable. The insert operator has no definite improvement result, but works in combination with the tour depletion operator.

4.2.2. Tour depletion Operator

This operation was added to ensure diversification and optimisation by removing a vehicle from the current solution. This will force the application to optimise without the specific vehicle if possible, else creating a new route.

4.2.3. *Relocate Operator*

This operation is mostly affected on optimising a current solution. Depending on the deterioration tolerance, it will move a stop from one route to another. The following graph shows that this operation does not have a high feasibility rate, even though the deterioration tolerance for the specific situation was not set. This means that any viable solution was acceptable to the problem, even if it results in a worse solution than the current best. What we can see from the graph is the ability after this operation to optimise.

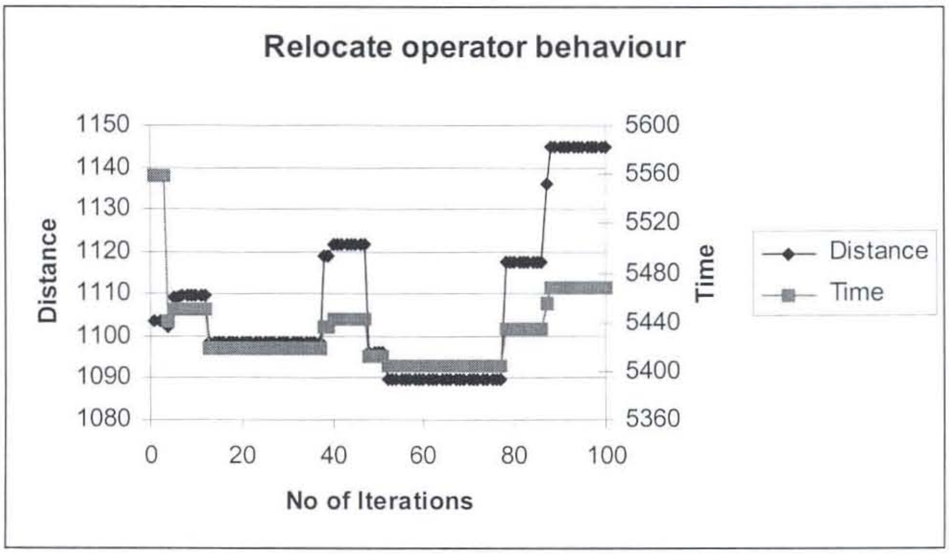

Figure 27: Relocate operator behaviour

4.2.4. *Exchange Operator*

The purpose of this operation is to swap two stops from different routes or within the same route with each other. The action can result in a better time utilisation or distance of the route. As can be seen from the graph below, this operation yields a feasible solution regularly. We can also see that the difference in the time or distance from the previous solution is not as big as with the relocate operator. The graph indicates that this operation is important to finding the local minimum.

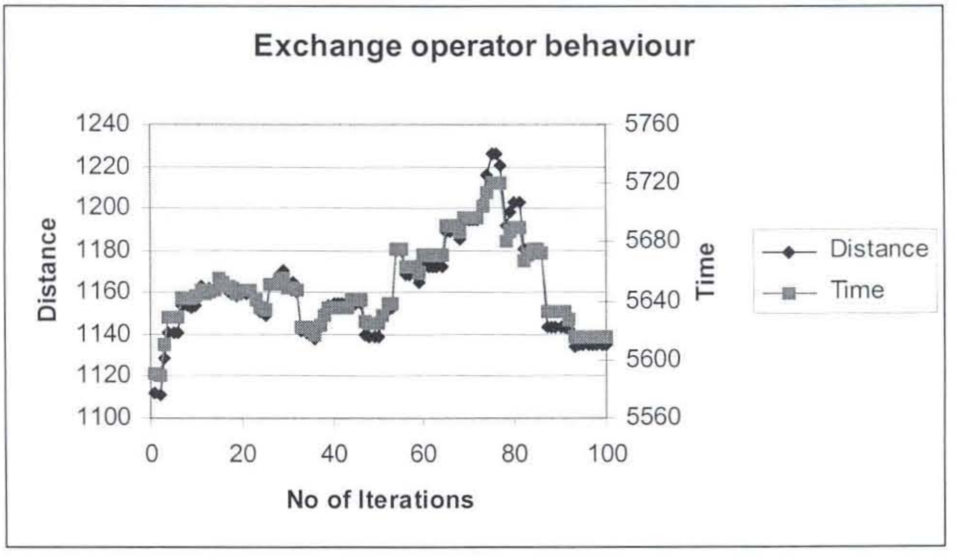

Figure 28: Exchange operator behaviour

4.2.5. 2-Operator

This operation takes two routes and cut them at specific positions and joins part one of route one with part two of route two and part two of route one with part one of route two. The implementation selects a target stop on route one and search for at a feasible swap route by traversing through its neighbours. As we can see from the graph, the move result in bigger changes from the previous solution, but has only a limited set of the viable moves. This can be seen in the latter part of the graph where the distance and time stays constant for long periods of iterations. We conclude that this is a result of the Tabu list that does not allow for previous moves to be repeated and no new moves exist.

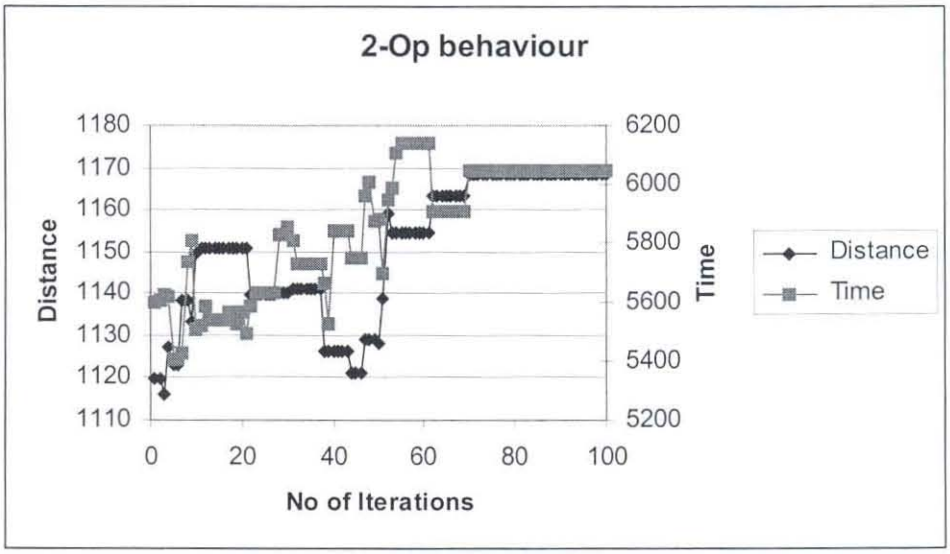

Figure 29: 2-Operator results

4.3. Application

In the previous paragraphs we showed the algorithm's performance with the 56 Solomon benchmark problems. This was done as a proof of concept for the algorithm. In this paragraph we will consider the result of a real life problem and show that the solution is feasible. The problem was taken from a commercial delivery company. All the variables were implemented as specified by the logistics manager.

Figure 30 shows the distribution of the stops as well as the solution. As stipulated in the initial research, the data environment is unpredictable. A quick analysis of the data indicates

- Inconsistent time window sizes.
- Random clustered stops.
- Long haul exceptions, relative to average stop distance from depot. The closest stop is less than 2 kilometres from the depot, while the furthest stop is more than 70 kilometres away.
- Some stops are located at the exact same position. From the figure we can make out some overlapping time windows.

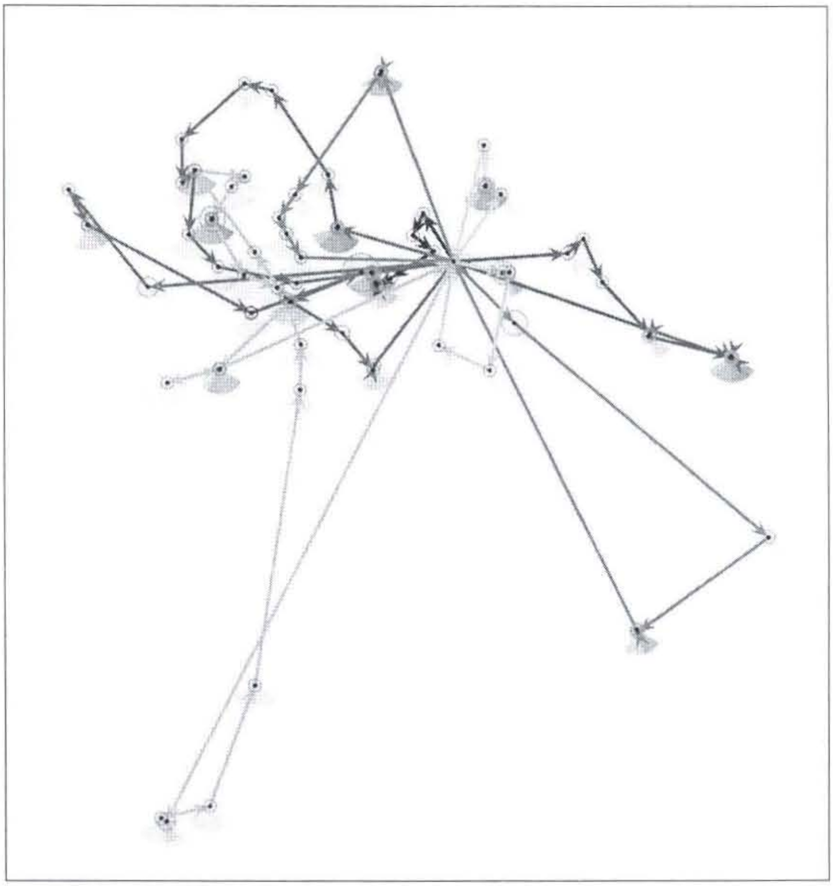

Figure 30: Application Solution 1

4.3.1. Initial Phase

From the results of initial solutions for Solomon's problems, we can conclude that using the latest departure time as criteria for a seed node will be sufficient. The following table shows the result of the initial solution on the real life problem. Because we are working with a heterogeneous fleet, all eight possible criteria have been implemented.

Initial Phase			
Criteria		Vehicles	Distance
Lowest Running Cost	Latest Departure	13	1251.92
	Earliest Arrival	12	1662.06
	Furthest	13	1259.07
	Closest	12	1186.82
Smallest Capacity	Latest Departure	13	1251.92
	Earliest Arrival	12	1522.54
	Furthest	13	1259.07
	Closest	12	1186.82

Table 8: Application Initial Phase

Although the latest departure criteria result in a comparative distance, the number of vehicles is higher than for the other methods. This confirms the decision to implement multiple criteria on the seed node selection.

4.3.2. Improvement

Figure 31 indicates the movement in the distance of the solutions for an execution of 5000 iterations. From the figure we can depict the ability of the algorithm to intensify and diversify.

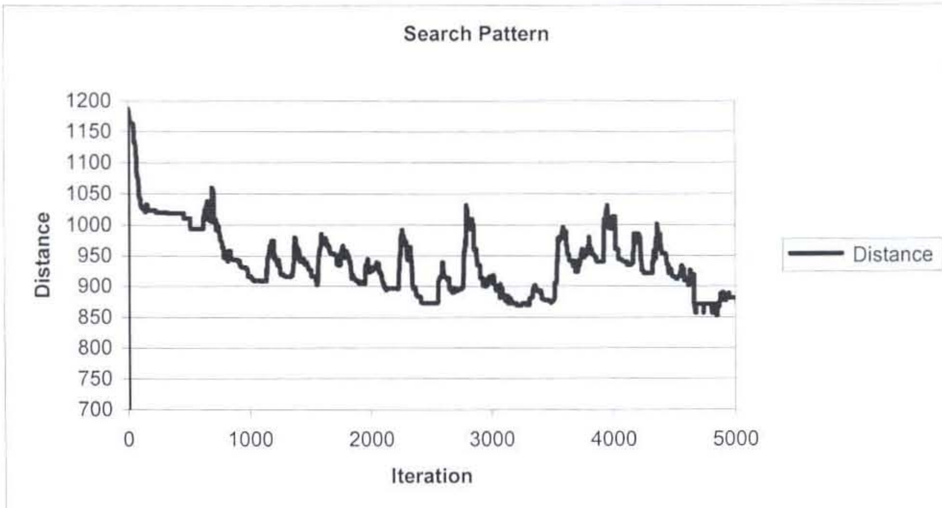

Figure 31: Search Pattern

The improvement heuristic started out with 12 vehicles and a distance of 1186 kilometres. After 5000 iterations we end up with 12 vehicles and a distance of 853 kilometres. This is an improvement of around 28% from the initial solution.

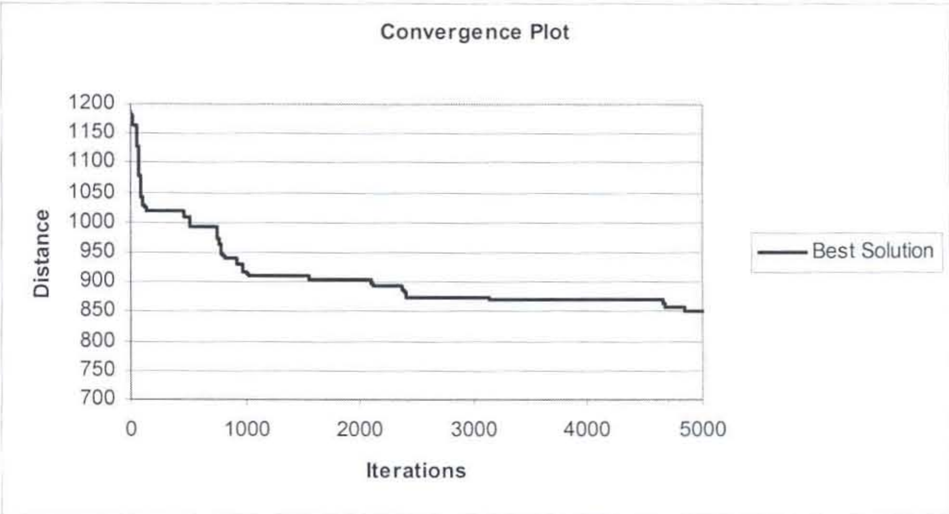

Figure 32: Convergence Plot

Another indication is the improvement in travel time. Travel time consist of the time it takes to travel between two stops depending on the time of the day. The travel time improved from 5789 minutes to 4497 minutes, an improvement of over 22% from the initial solution.

We also implemented multiple operations to move from a current solution to a valid neighbour solution. By keeping track of the success of an operation, we statistically balance the random selection of an operation. This technique results in the use of a better combination of operations depending on the data distribution and constraints of the problem instance. A hybrid with the Simulated Annealing method allows the solution to diversify and intensify periodically, while keeping track of moves through Tabu lists. Figure 31 indicates the ability of the algorithm to achieve this goal.

Figure 32 shows the ability of the algorithm to converge. We tested the 56 Solomon benchmark problems to indicate the validity of the algorithm. Solomon's problem is a simple instance of the problem we consider, but there does not exist benchmark problems for our set of problems. Table 5 shows that the new algorithm is effective on Solomon's benchmark problems.

The results prove that the implemented algorithm is effective to solve the set of problems encountered in an ASP environment. With the knowledge gained, we can continue to search for new operations and methods to improve the efficiency of the algorithm in the generic ASP environment.