5 ENVIRONMENT ANALYSIS - RESULTS

5.1 Overview

This chapter discuss examples of the results achieved from specific methods selected in the study. It explains the environments and compares results of different methods for the different environments.

The results are based on the implementation of Solomon's cost and constraint function. This implementation is sufficient to prove other implementations on the adaptive object model. The next paragraph defines our implementation method for the Solomon cost functions.

5.2 Solomon Functions

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.

The implementation requires us to define the domain model as well as the cost and constraint functions for Solomon. Solomon implements a basic VRPTW problem. The following diagram represents the implemented stop class used in the solution.

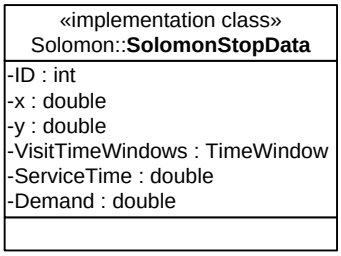

Figure 38: Solomon Domain Instance

Solomon constraints include the following checks:

- Time Windows – each stop has an open and close time window in which the stop must be services. Time windows create a *wait time* that must be considered in the cost.
- Volume – there exist a capacity constraint on each vehicle used in the solution. The volume creates a *service time* at a stop which can be considered at the cost function.

Distance in Solomon does not have a constraint, but to improve cost calculation, distance changes can be updated during a constraint check as constraints are called with a special instruction. Each *routestop* implements an abstract of the *RouteStopData* class which acts as special container for specific implementations. The Solomon implementation can be defined as follows:

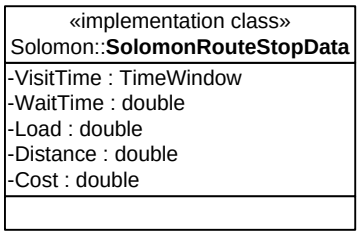

Figure 39: Solomon *RouteStopData* implementation

This adaptive implementation class extends the algorithm memory which contributes to efficiency. The effect of the implementation depends on the implementer's knowledge of the problem domain. The attributes of this Solomon implementation class is as follows:

- Visit Time – the arrival and departure time of the vehicle at the specific route stop for the specified route.
- Wait Time – the time a vehicle is too early and has to wait for the closest forward opening time to occur.
- Load – the total load on the vehicle for the route to the specific stop.
- Distance – the total distance travelled for the route to the specific stop.
- Cost – the total cost incurred for the route to the specific stop.

The efficiency of the adaptive model can now be explored. These structures allow the operator to implement effective methods for calculating cost and checking constraints. The knowledge the operator has of the problem environment can assist in implementing another level of meta heuristic. The basic rules followed in this thesis for Solomon is as follows:

1. Changes to a route on the stop list impacts only attributes of route stops after the stop.
2. Solomon use Euclidean distance and time which comply to the triangular rule:

$$d_{ij} \leq d_{ik} + d_{kj}\,.$$

5.3 Probability matrix

The probability matrix is an important memory structure which guides the algorithm. The analysis of the environment is used to assist in the evaluation of the surface chart shape of possible neighbouring solutions. Constraints such as time windows contribute to the shape of possible solutions. The results display some common patterns that exist in the benchmark problems.

	Depot	Stop 1	Stop 2	Stop 3	Stop 4	Stop 5	Stop 6	Stop 7	Stop 8
# Neighbours from stop	100	3	9	97	15	98	29	75	68
Sum of cost	2885	76	226	3109	368	3106	874	2443	2295
$min\ P(x_{ij})$	11%	88%	83%	13%	78%	12%	67%	31%	36%
$max\ P(x_{ij})$	90%	90%	90%	90%	90%	90%	90%	90%	90%
# Neighbours to stop	100	93	88	15	80	2	76	21	32
Sum of cost	2885	3133	3061	410	2749	20	2622	620	1018
$min\ P(x_{ij})$	54%	40%	38%	42%	40%	70%	39%	42%	40%
$max\ P(x_{ij})$	120%	90%	90%	90%	90%	90%	90%	89%	90%

Table 5: C105 Statistic summary extract

The first column indicates that each stop is reachable from the depot. If this was not true, we were faced with an infeasible solution space. The average cost per stop can be used as benchmark for other stops. The rest of the columns represent the statistics per stop. The extract in Table 5 is only for stop 1 to 8 in Solomon's C105 problem. The summary provides an indication of the influence of constraints on the problem and it can be clearly deducted that constraints effect varies from stop to stop.

Stop 1 has the definite ability to only go to 3 other stops, while 93 other stops can have stop 1 as a subsequent neighbour. The probability distribution between the 3 possible subsequent stops is between 88% and 90%, which can be seen as high enough to know that the likelihood of all of the 3 stops to be the neighbour in the final result. These values are used in the pheromone trial. The statistic can immediately be applied on the initial pheromone trial.

The significant difference between from and to stop statistics indicate on a reduced number of good solutions that exist. It supports the intuition that a good or even best solution can be

reach through a limited number of moves. The statistics does not reveal anything clear on groups of stops, and is used in guiding the improvement stage.

	Depot	Stop 1	Stop 2	Stop 3	Stop 4	Stop 5	Stop 6	Stop 7	Stop 8
# Neighbours from stop	100	31	99	56	34	99	70	86	75
Sum of cost	2495	849	2919	1696	1078	2920	1834	2618	2549
$min\ P(x_{ij})$	11%	65%	12%	46%	63%	12%	35%	22%	31%
$max\ P(x_{ij})$	90%	90%	90%	90%	90%	90%	90%	90%	90%
# Neighbours to stop	100	99	48	84	99	38	77	66	73
Sum of cost	2495	2832	1407	2774	3433	1065	1993	1959	2347
$min\ P(x_{ij})$	52%	45%	37%	34%	30%	36%	42%	38%	31%
$max\ P(x_{ij})$	93%	90%	87%	90%	89%	87%	90%	89%	90%

Table 6: R205 Statistic summary extract

The statistics predicts what we already know from the domain environment of the R205 problem. The number of stops from and stops to as neighbours are much more evenly spaced, which indicates more options of combinations and thus a bit more difficult to deduct an optimal solution from limited moves. The difference between the minimum and maximum probability does however indicate that quality of combination can be used to influence decision making on moves. The pheromone trial can immediately reflect these properties.

	Depot	Stop 1	Stop 2	Stop 3	Stop 4	Stop 5	Stop 6	Stop 7	Stop 8
# Neighbours from stop	75	74	74	74	74	74	74	74	74
Sum of cost	1815	2343	1933	2242	1830	2325	1856	2118	2140
$min\ P(x_{ij})$	11%	12%	12%	12%	12%	12%	12%	12%	12%
$max\ P(x_{ij})$	90%	90%	90%	90%	90%	90%	90%	90%	90%
# Neighbours to stop	75	74	74	74	74	74	74	74	74
Sum of cost	1815	2343	1933	2242	1830	2325	1856	2118	2140
$min\ P(x_{ij})$	53%	31%	43%	37%	49%	31%	49%	38%	37%
$max\ P(x_{ij})$	93%	89%	90%	90%	88%	89%	90%	90%	90%

Table 7: P n76 k4 Statistic summary extract

The statistics in Table 7 indicate a problem space where constraints do not play any role in the preliminary evaluation of the environment. This can be seen from the equal number of possible from and to stops. The extract only show up to stop 8, but all stops showed equal

result for number of possible from and to stops. A stop is the neighbour of every stop in the problem space, except itself. If we look at the domain definition of this problem, it consists of 75 stops and can be optimally solved with 4 vehicles. There is no time constraint on the problem and therefore all combinations of stops are possible. Capacity constraints' impact can only be seen when building a route.

Our first step in solving the problem proved to indicate some trends in the environment. We compared the computed results with the common domain knowledge, and the indication provided by the results is clear enough to use as guidance for the algorithm. We can now add more information through other methods.

5.4 Density cluster results

The following section explains the advantage of density clustering on certain problem environments. Density clustering assists in environments where natural groups exist. In a real-life environment, locations can be grouped in residential areas that are split by commercial properties, nature, roads, etc. Clustering data can assist to generate good solutions quick and effective.

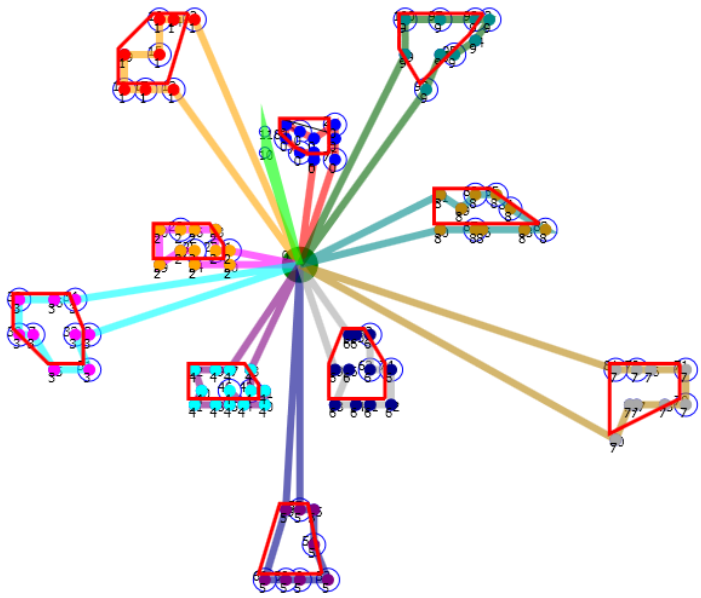

Figure 40: C105 SIH on density cluster

Problem class C1 is specifically designed to benchmark solution algorithms against clustered solutions. The image in Figure 40 is the result of the best initial solution from the algorithm. It is clear that the density clusters forced the initial algorithm into a near optimal solution. The red polygon areas indicate the clusters build from the probability matrix that was constructed in the solution preparation phase. One of the aims for this study is also to provide alternative solutions which might not be theoretical the best, but can indicate to the logistic manager what other possibilities exist. This initial result indicates that alternative solutions that have to be dissimilar will be tough to achieve.

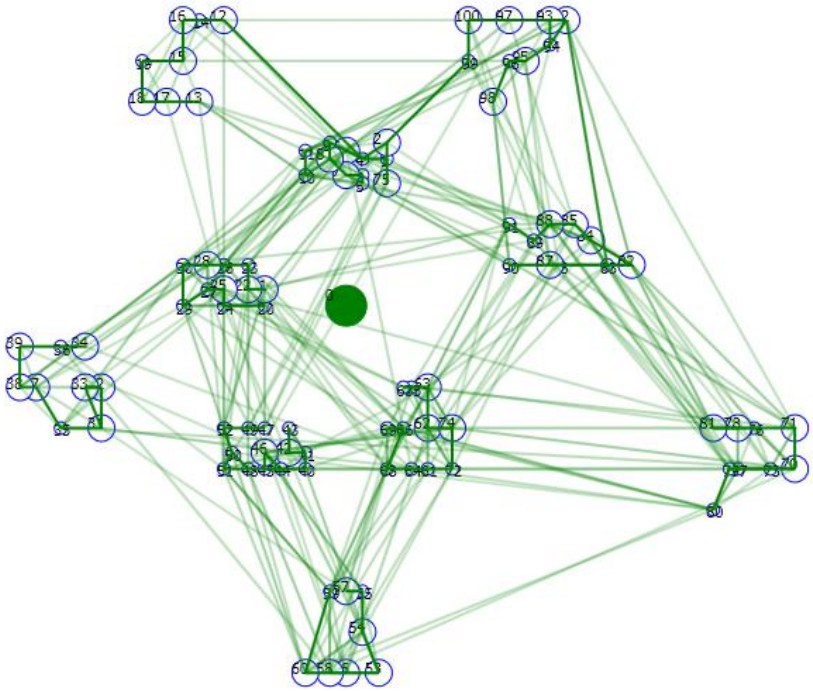

Figure 41: C105 initial pheromone trial

The initial solution algorithms consist of more than just the clustered approach. To ensure that anomalies are also catered for in extreme cases, the pheromone trial is still build up from other results as well. Figure 41 represents the initial pheromone trial before the improvement phase starts. It is clear from comparing with Figure 40 that the clustered stop segments have been given a significant higher probability.

Density clustering is a very effective method for specific cases. Because the method relies on specific criteria to generate a cluster, some environments can result in none or few clusters. Although it still assists in guiding the pheromone trial, we continue to apply additional methods in searching for guidance.

5.5 Partition cluster results

Clusters generated by a partition algorithm do not necessarily reflect a strong binding between the elements as with density clusters. Partition clustering is therefore only used when no or a small number of density clusters exist. The partition cluster method requires a predefined number of clusters to be generated and does not guarantee the same result if the algorithm is executed again.

The use of partition cluster is to accelerate the convergence to a good feasible solution. The number of clusters is not defined as the expected final number of routes, but rather large enough to generate chains to work with.

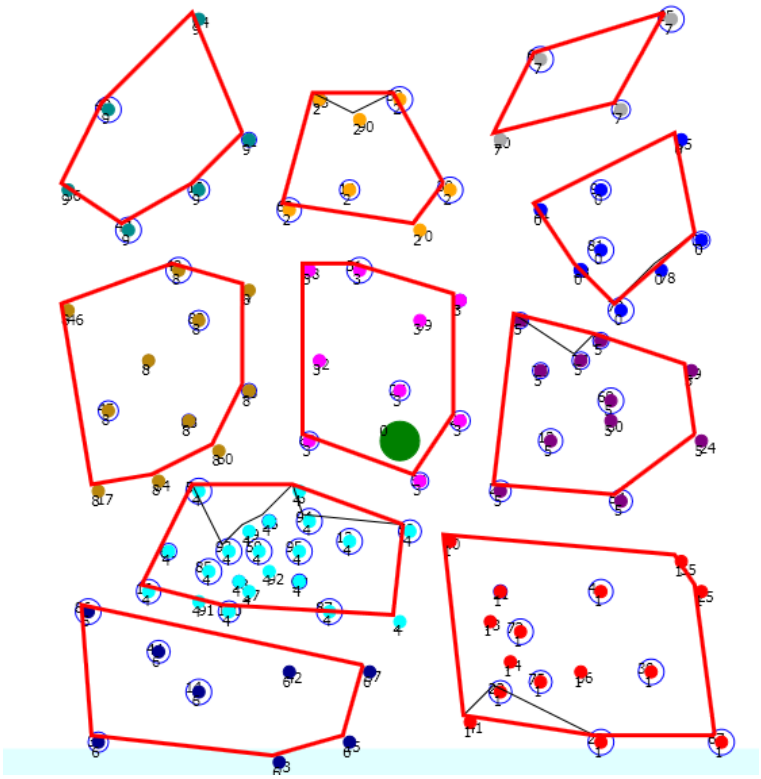

Figure 42: R108 Partition cluster

The R108 Solomon benchmark problem is a random generated environment with a short scheduling horizon. The partition clustering method is non-deterministic algorithm which allows multiple outcomes. Clustering still reflects something from the environment.

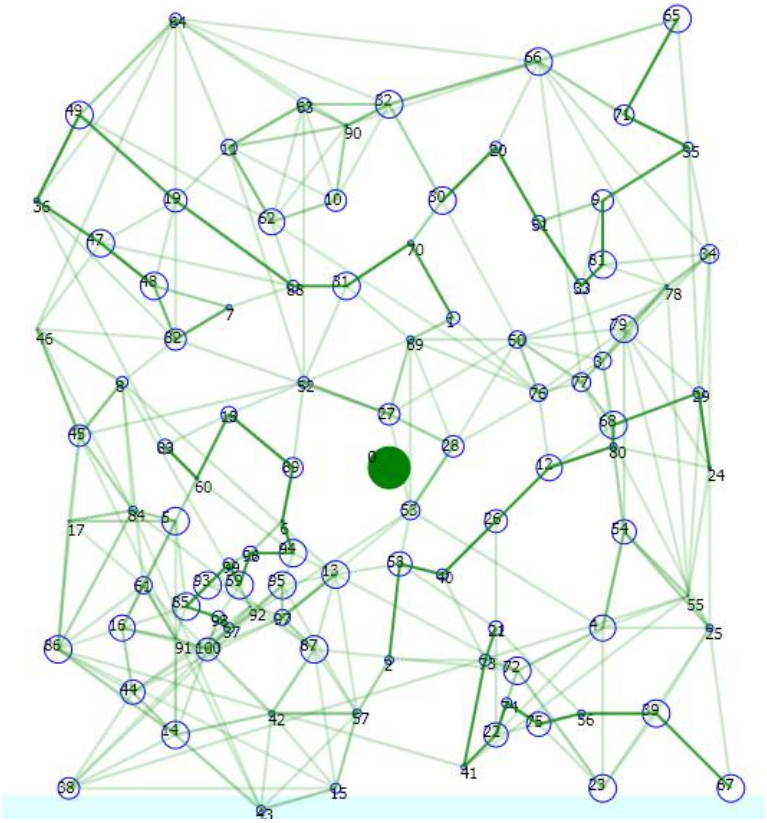

Figure 43: R108 Initial pheromone trial

From the pheromone trial in Figure 43, it is clear that the partition clustering did not affect the initial trial as much as the density method. This is due to the contribution of other factors that did not correlate with all the cluster chains.

5.6 Benchmark Results

This paragraph lists the execution results on some benchmark problems. The main focus is on Solomon's problems. Other examples use the same cost calculation method of Solomon. Executing the algorithm on Solomon's benchmark problems returned very good results.

The best results are from Sintef's website (Solomon-benchmark, 100-customers, 2010) . The tables represent the

- Problem – e.g. C101, RC104, etc.
- Initial Solution – according to a PBIH (Number of vehicles and distance travelled, Moolman (2004))
- Tabu improvement applied (Number of vehicles and distance travelled, Moolman (2004))
- Initial Solution – according this multi-start method (Number of vehicles and distance travelled)
- Improvement applied (Number of vehicles and distance travelled)
- Best published (Number of vehicles, distance travelled and % difference of the study's best) – from Sintef website (Solomon-benchmark, 100-customers, 2010)

5.6.1 C1

The following diagram (Figure 44) display the solution of Solomon's C101 problem. The left diagram is the final routes and the right diagram is the final pheromone trail. The total route distance for the solution is 828.94, which match the best published solution. The clear pheromone trial indicates that alternatives to these routes are not feasible to consider.

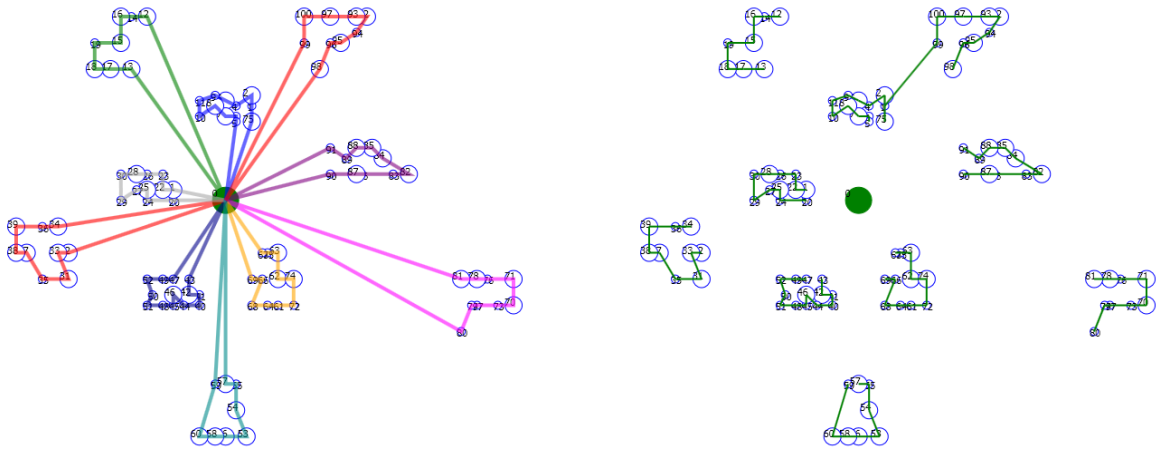

Figure 44: Solomon C101 Solution

The following diagram (Figure 45) display the overlay of the initial clusters overlaid with the solution. The red polygon and same colour stops indicate a cluster. Although the C101 is a

simple solution, it indicates that the cluster should influence the initial pheromone trail as well as Tabu List.

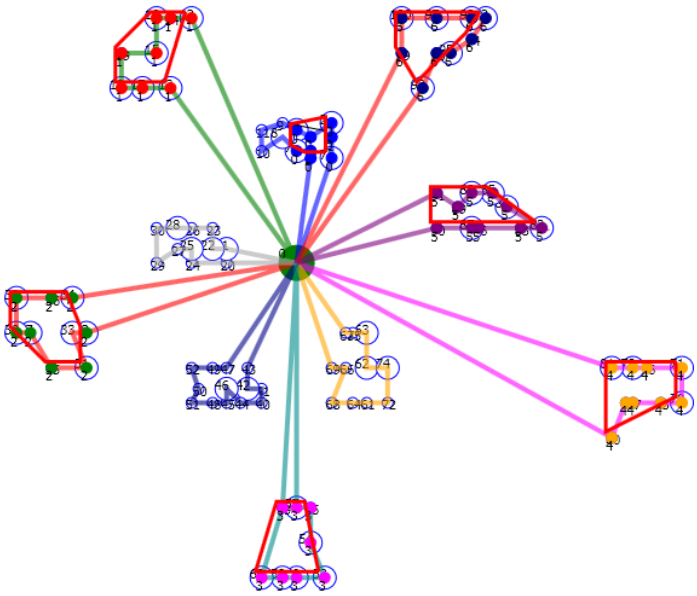

Figure 45: Solomon C101 Cluster overlay

The following diagram (Figure 46) display the solution of Solomon's C109 problem. The total route distance for the solution is 828.94, which match the best published solution.

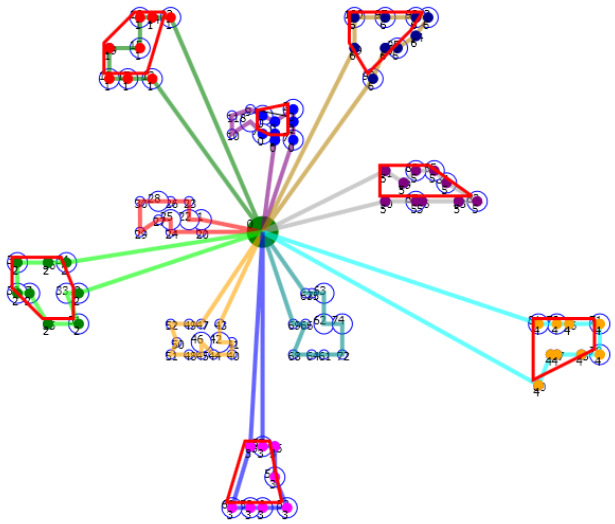

Figure 46: Solomon C109 solution

Table 8 represents the solutions compared to a plain Tabu solution, the best initial solution, the adaptive object algorithm and the best published solution.

			Tabu		New					
Prob	Initial Solution		Improvement		Initial Solution		Improvement		Best Published	
C101	10	880.47	10	828.94	11	860.58	10	828.94	3.7%	10	828.94	**0.0%**
C102	10	997.74	10	871.32	16	1,144.04	10	828.94	27.5%	10	828.94	**0.0%**
C103	10	1081.56	10	916.83	16	1,237.34	10	853.72	31.0%	10	828.06	**3.1%**
C104	10	1059.59	10	911.85	12	975.23	10	836.84	14.2%	10	824.78	**1.5%**
C105	10	878.78	10	827.55	11	860.58	10	828.94	3.7%	10	828.94	**0.0%**
C106	10	968.58	10	840.19	13	976.02	10	834.23	14.5%	10	828.94	**0.6%**
C107	10	928.74	10	827.55	11	860.58	10	834.23	3.1%	10	828.94	**0.6%**
C108	10	871.57	10	827.55	12	910.35	10	828.94	8.9%	10	828.94	**0.0%**
C109	10	910.28	10	829.74	13	979.95	10	850.26	13.2%	10	828.94	**2.6%**

Table 8: C1 Solutions

5.6.2 C2

The setup of the cost functions in the adaptive object model in this contains no cost for the usage of an extra vehicle. This can already be seen by the result of the best initial solution from the multi-start approach. It is assumed that the traditional SIH best solution is contained in the set of initial solutions, but is not reflected as the best. That is evident by comparing the

original initial solution with the new multi-start result. The new algorithm did fairly well on the optimisation for C2 type problems.

	Tabu				New						
Prob	Initial Solution		Improvement		Initial Solution		Improvement			Best Published	
C201	3	826.15	3	588.88	5	673.69	3	591.56	12.2%	3	591.56	0.0%
C202	3	1180.34	3	623.46	6	720.65	4	619.44	14.0%	3	591.56	4.7%
C203	3	1173.25	3	625.46	4	838.94	4	617.20	26.4%	3	591.17	4.4%
C204	3	1235.70	3	685.10	5	870.92	3	646.54	25.8%	3	590.60	9.5%
C205	3	789.79	3	617.45	4	777.30	3	589.72	24.1%	3	588.88	0.1%
C206	3	934.87	3	629.63	5	875.20	3	591.35	32.4%	3	588.49	0.5%
C207	3	884.44	3	587.89	5	869.49	3	601.92	30.8%	3	588.29	2.3%
C208	3	815.97	3	592.93	5	759.59	3	594.73	21.7%	3	588.32	1.1%

Table 9: C2 Solutions

5.6.3 R1

The effect of the randomly distributed stops in the R1 environment result in the inefficient use of the clustering approach for the initial solution. The optimisation produced good results and we can argue that the grouping during the initial solution assisted to guide the meta-heuristic in the correct areas. The greedy nature of the initial solutions resulted in unordered stops, but the heuristic's swap moves improved the result with a low computing cost.

	Tabu				New						
Prob	Initial Solution		Improvement		Initial Solution		Improvement			Best Published	
R101	20	1857.93	20	1,670.13	25	2264.223	21	1673.5645	26.1%	19	1,645.79	1.7%
R102	19	1792.59	19	1,576.81	23	2141.259	19	1491.2923	30.4%	17	1,486.12	0.3%
R103	15	1553.58	15	1,316.31	16	1651.6	15	1245.2884	24.6%	13	1,292.68	-3.7%
R104	12	1283.22	11	1,061.90	14	1419.203	12	1027.7393	27.6%	9	1,007.24	2.0%
R105	15	1534.40	15	1,455.08	22	1844.607	16	1401.7127	24.0%	14	1,377.11	1.8%
R106	15	1457.51	14	1,292.28	17	1767.745	14	1280.718	27.6%	12	1,251.98	2.3%
R107	13	1336.79	12	1,174.00	15	1584.442	12	1130.3644	28.7%	10	1,104.66	2.3%
R108	10	1174.06	9	1,030.87	12	1313.324	12	1006.2199	23.4%	9	960.88	4.7%
R109	14	1423.01	13	1,284.32	19	1714.363	14	1189.11	30.6%	11	1,194.73	-0.5%
R110	12	1332.66	13	1,205.48	17	1562.29	13	1136.9781	27.2%	10	1,118.59	1.6%
R111	13	1344.17	13	1,239.26	14	1604.695	13	1112.9772	30.6%	10	1,096.72	1.5%
R112	11	1167.79	11	1,059.78	12	1302.659	11	1001.0385	23.2%	9	982.14	1.9%

Table 10: R1 Solutions

5.6.4 R2

The environment in R2 which consist of a combination of random stops and low capacity constraint effect, resulted in better results through the use of more vehicles. The initial solution also display the trend to use more vehicles to get a shorter distance travelled.

	Tabu				New					
Prob	Initial Solution		Improvement		Initial Solution		Improvement			Best Published	
R201	5	1633.98	4	1,335.55	8	1,535.61	8	1,172.01	23.7%	4 1,252.37	-6.4%
R202	5	1570.04	4	1,200.26	8	1,363.80	8	1,102.86	19.1%	3 1,191.70	-7.5%
R203	4	1325.15	3	972.59	7	1,280.42	7	934.83	27.0%	3 939.54	-0.5%
R204	3	1054.39	3	842.54	6	1,075.00	5	771.54	28.2%	2 825.52	-6.5%
R205	4	1461.61	3	1,133.02	5	1,396.79	7	1,023.90	26.7%	3 994.42	3.0%
R206	3	1358.68	3	985.94	6	1,221.69	5	974.70	20.2%	3 906.14	7.6%
R207	3	1205.44	3	948.50	6	1,117.92	5	839.83	24.9%	2 893.33	-6.0%
R208	3	908.49	2	845.94	6	973.85	4	845.47	13.2%	2 726.75	16.3%
R209	4	1260.75	4	930.43	6	1,160.23	6	909.86	21.6%	3 909.16	0.1%
R210	4	1384.77	3	1,019.45	6	1,220.80	7	948.19	22.3%	3 939.34	0.9%
R211	3	1080.89	3	862.42	5	1,133.56	5	838.96	26.0%	2 892.71	-6.0%

Table 11: R2 Solutions

5.6.5 RC1

The RC problems might be the closest representation to real world environments. It represents an environment that is not predictable and that contains areas with patterns in the environment. The algorithm will benefit from the clustered areas and the pheromone trail that is influenced for used during improvement will assist in a balanced diversification strategy. Results with the cost function utilised more vehicles in certain problems which produced better results.

	Tabu				New					
Prob	Initial Solution		Improvement		Initial Solution		Improvement			Best Published	
RC101	16	1929.02	16	1,742.62	23	2,159.95	17	1,704.72	21.1%	14 1,696.94	0.5%
RC102	15	1789.29	15	1,625.30	23	1,985.22	15	1,535.30	22.7%	12 1,554.75	-1.3%
RC103	13	1613.99	13	1,403.99	18	1,790.02	12	1,327.32	25.8%	11 1,261.67	5.2%
RC104	12	1363.74	12	1,212.92	15	1,506.09	12	1,212.79	19.5%	10 1,135.48	6.8%
RC105	16	1805.33	16	1,706.53	21	2,040.03	17	1,576.19	22.7%	13 1,629.44	-3.3%
RC106	14	1581.39	14	1,502.00	17	1,873.54	14	1,483.42	20.8%	11 1,424.73	4.1%
RC107	13	1607.96	12	1,318.22	15	1,728.79	12	1,300.15	24.8%	11 1,230.48	5.7%
RC108	12	1340.10	12	1,240.27	13	1,438.51	11	1,164.81	19.0%	10 1,139.82	2.2%

Table 12: RC1 Solutions

5.6.6 RC2

The RC2 problems display the same trend as with the R2 problem, mainly because of the use of more vehicles. The multi-start initial approach allows best solutions to use more vehicles than the best registered initial solution. Most of the solution results display better than published because of the use of the vehicles.

			Tabu		New				
Prob	Initial Solution		Improvement		Initial Solution		Improvement		Best Published		
RC201	5	2131.14	4	1,474.86	9	1,856.60	8	1,317.55	29.0%	4	1,406.91	-6.4%
RC202	5	1943.42	4	1,298.28	10	1,700.06	8	1,148.13	32.5%	3	1,367.09	-16%
RC203	4	1595.85	3	1,081.34	8	1,534.76	7	985.91	35.8%	3	1,049.62	-6.1%
RC204	3	1184.48	3	883.53	5	1,107.90	5	829.57	25.1%	3	798.41	3.9%
RC205	6	1940.44	5	1,311.93	8	1,766.76	9	1,180.35	33.2%	4	1,297.19	-9.0%
RC206	4	1595.74	4	1,162.03	8	1,480.07	7	1,121.70	24.2%	3	1,146.32	-2.1%
RC207	4	1491.13	4	1,106.24	6	1,607.20	6	1,054.30	34.4%	3	1,061.14	-0.6%
RC208	3	1275.21	3	920.17	5	1,106.80	4	858.56	22.4%	3	828.14	3.7%

Table 13: RC2 Solutions

5.7 Summary

The result indicates that the algorithm is giving consistent results across all the different problems. The implementation of the Solomon cost function depends solely on the distance, which resulted in answers that has beaten the best published.

The number of vehicles used is more in these instances. The function has been left in this state to indicate the sensitivity of the algorithm on the adaptive object model. i.e. the implementation of the cost and constraint classes by the implementer. This flexibility cannot be achieved in a domain orientated implementation.

The flexibility of the solution can now be exploited and by altering only the cost function to incorporate the cost of the use of another vehicle.

6 PARALLEL IMPLEMENTATION

6.1 Overview

This chapter provides a brief overview of the adaption of the ASAO algorithm for parallel use. The objective is mainly to gain speed improvement with the least complexity in the adaption. The initiation of the approach will assist in alternatives for future expansion.

The design of a parallel algorithm can be viewed as consisting of four stages - Partitioning, Communication Analysis, Granularity Control and Mapping. The following simple example illustrates some of the issues involved in each of the stages.

Consider the scenario: *n* answer-scripts have to be marked, each of which contains the answers to *m* questions. The scripts could be viewed as the data, and the marking process itself as the computation to be performed on it.

In order to design a parallel solution to this problem, it must first be decomposed into smaller tasks which can be executed simultaneously. This is referred to as the partitioning stage.

This can be done in one of two ways. Each script could be marked by a different marker - this would require *n* markers. Alternatively, marking each question could be viewed as a task. This would result in *m* such tasks, each of which could be tackled by a separate marker, implying that every script passes through every marker.

In the first approach, the data (scripts) is first decomposed and then the computation (marking) is associated with it. This technique is called domain decomposition.

In the second approach, the computation to be performed (marking) is first decomposed and then the data (scripts) is associated with it. This technique is called functional decomposition. The partitioning technique that will be chosen often depends on the nature of the problem.

Suppose one needs to compute the average mark of the n scripts. If domain decomposition was chosen, then the marks from each of the markers would be required. If the markers are at different physical locations, then some form of communication is needed, in order to obtain the sum of the marks.

The nature of the information flow is specified in the communication analysis stage of the design. In this case, each marker can proceed independently and communicate the marks at the end. However, other situations would require communication between two concurrent tasks before computation can proceed.

It may be the case that the time to communicate the marks between two markers is much greater than the time to mark a question. In which case, it is more efficient to reduce the number of markers and have a marker work on a number of scripts, thereby decreasing the amount of communication.

Effectively, several small tasks are combined to produce larger ones, which results in a more efficient solution. This is called granularity control. For example, k markers could mark n/k scripts each. The problem here is to determine the best value of k.

The mapping stage specifies where each task is to execute. In this example, all tasks are of equal size and the communication is uniform, so any task can be mapped to any marker. However, in more complex situations, mapping strategies may not be obvious, requiring the use of more sophisticated techniques.

6.2 Single thread environment

The implementation of the ASAO algorithm was done from basic principles in the C# language. A standard off the shelf desktop was used to do the development and testing. The most basic desktops contain multiple processors. This machine runs two independent processor cores in one physical package at the same frequency.

With the introduction of the multi-processors, and Hyper-Threading Technology for the desktop, threading is no longer within the exclusive domain of Server application developers. In a single-core or traditional processor the CPU is fed strings of instructions it must order, execute, then selectively store in its cache for quick retrieval. When data outside the cache is required, it is retrieved through the system bus from random access memory (RAM) or from storage devices. Accessing these slows down performance to the maximum speed the bus, RAM or storage device will allow, which is far slower than the speed of the CPU. The situation is compounded when multi-tasking. In this case the processor must switch back and forth between two or more sets of data streams and programs. CPU resources are depleted and performance suffers.

In a dual core processor each core handles incoming data strings simultaneously to improve efficiency. Just as two heads are better than one, so are two hands. Now when one is executing the other can be accessing the system bus or executing its own code.

To utilize a dual core processor, the operating system must be able to recognize multi-threading and the software must have simultaneous multi-threading technology (SMT) written into its code. SMT enables parallel multi-threading wherein the cores are served multi-threaded instructions in parallel. Without SMT the software will only recognize one core. The CPU usage in Figure 47 displays the normal implementation of the ASAO algorithm in a Windows environment.

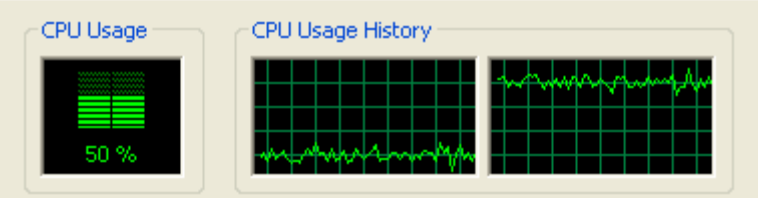

Figure 47: Desktop CPU usage single thread

The usage is clearly showing that the operating system does not balance processing across the two processors. The responsibility to implement a parallel solution is that of the developer. The following paragraphs step through the thinking of implementing the parallel ASAO.

6.3 Partitioning

Decomposing the algorithm into smaller tasks require an evaluation of the steps followed. We investigate the steps from a high level.

The first high-level step is reading the problem environment. This step requires basic information management and although there exist methods of improving reading data, we will not consider it as part of this design.

The second step consists of pre-calculations such as the calculation of the probability matrix and detection of neighbours. This still not part of the core algorithm and is done as a once off, but can easily be done in parallel as it is two non dependent separate processes. This is not currently of interest for us.

The third step is defined as the environment evaluation. This procedure is just a statistical review of the environment and does not contribute to the running time of the algorithm. It must be noted that when moving to a dynamic environment, this step might be an important tool in the continuous evaluation of the environment and can contribute to the effectiveness of the algorithm. It is not part of this study.

The core of the algorithm consists of two main steps, the initial solution and the improvement.

The initial solution approach is designed to provide multiple starting options through the use of multiple methods. This assists to diversify into the problemspace which assist in adapting to the environment sensitivity according to a specific method. In other implementations, an initial solution method is chosen depending on the environment and the improvement kicks-off from a single solution start. This lead to the initial solution being designed for the specific problem environment. The initial solution has been disregarded in the overall computational time. The impact of the initial solution approach in this study is significant enough to consider parallel processing. We define the parts as each different method that is used.

The improvement algorithm is the most complex to consider because of the impact it has on the computational time. The dependence on the number of ants to solve the process influences the efficiency of the parallel implementation. The ASAO algorithm provides a clear location for the partitioning, i.e. the individual ants that is responsible for the implementation of an improvement technique.

The challenge is to consolidate the results on the correct times and have the impact varies depending on the solution overlap and solution status.

6.4 Communication Analysis

It is clear from the previous paragraph that we will only consider the initial and improvement part of the solution for parallel implementation. This paragraph analyzes the communication required between the defined parts of the algorithm. The functional decomposition technique is used because we cannot clearly define the tasks inside the computation. The heuristic approach is dependent on randomness.

The initial solution approach spawns multiple scouts into the problem space to search for possible solutions. Scouts act individually and do not communicate with each other during the search process. They communicate their findings back to the queen who has to make a decision on the next action. The communication from the scouts to the queen contains all the solutions per scout and the method of the scout.

The queen can now evaluate the difference in quality for each scout and if necessary combine subparts from solutions to create new solutions. The communication required for the scouting does not depend on concurrent processes to share information.

The improvement phase's efficiency is more susceptible to communication between the processes. The heuristic type algorithm depends on memory structures to assist in decision making. In the single thread environment, the sequence can be determined and information can be passed over to the next ant on termination. Although complex, the implementation of parallel processing can contribute in speed of execution. With intelligent design, the speed can even be more approved by reducing the number of iterations because of communication between concurrent processes.

In the single thread environment, a process will finish before information is passed to the next action. If two processes based on different actions were executing concurrently, frequent messaging can abort the process that is either not efficient or running in the same area of the problem space.

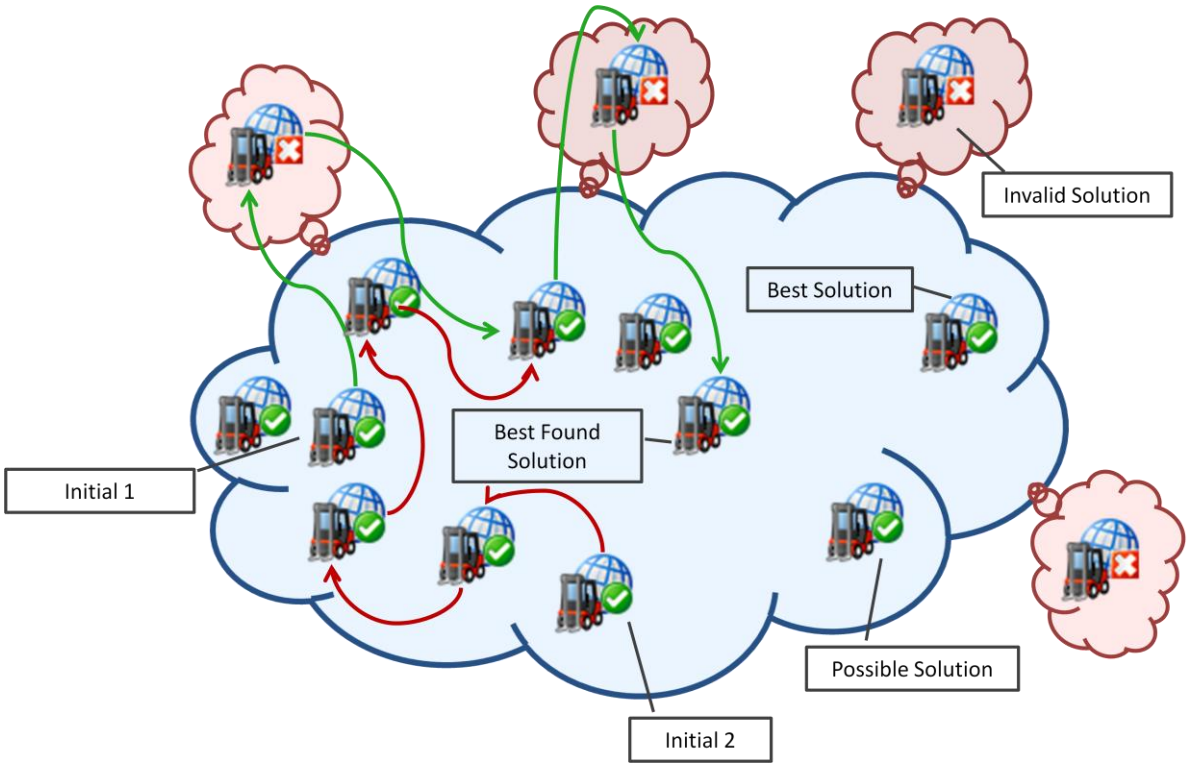

Figure 48: Multiple search paths

The scenario in Figure 48 depicts a high-level search pattern of the algorithm. The green line represents process 1 and the red line process 2. It is clear that the two processes are traversing in the same area of the problem space. The effort could be better spend on the area on the right hand side.

This is the main reason why traditional heuristic methods include a step for diversification. And the solution would still be found in traditional heuristic methods, but the time waist could have been detected earlier.

The proposed approach sounds simple enough, but the effort of the communication and comparing solutions at run-time is the major factors that will determine the success of this step. The environment knowledge must now play its part.

If the environment's topology is flat, as described in chapter 5, we know that there exist multiple solutions in a neighbourhood of moves. This environment normally requires a wide search and intensive local optimization. One process in a specific area is sufficient for the convergence because the local memory structures will guide the optimization the best. Comparing search patterns in this environment is difficult because solutions that exist in the

same area can have constructed totally different route structures. All because there exist a number of solutions in the neighbourhood that can be reached by one move.

If the environment topology consists of more mountains and valleys, the local optimization tends to converge to a locally optimal quite easily. The goal would be to terminate those processes that are climbing the same hill and only leave one to achieve the goal. Detecting these similarities should be easier in this environment, as the first few steps would quickly settle on a pattern of routes.

The proposed implementation of the communication channels consist of adding an additional layer of communication. The queen, who is in control, should place all ants in groups determined from the initial solutions. Efficiency can then be compared by

- Convergence speed – how does the algorithm climb the hill in relation to other processes? If the process is lacking in convergence compared to total cost, the area is most likely not suitable for a best solution.
- Route overlapping – stop sequence change and individual stop exchange between routes are the most effective moves close to the local optimum. Routes from the same area would overlap with a high percentage.
- Total cost – if the total cost of a process is not considered to be feasible, terminate.

To add to the efficiency of the communication, the importance of the efficiency parameter result should depend on the global status of the solution. The hybrid implementation based on simulated annealing provides information on the status of the solution.

In the initial phases of the improvement heuristic, convergence speed will be the dominant factor. The principle of initial solutions, that might be costly but still good because of its convergence in a specific area, applies. Total solution cost is always an important factor, while the route overlapping factor becomes dominant in the second part of the cooling down process.

6.5 Granularity Control

Determining the granularity of the implementation is not a straight forward task. The initial solution approach follows a strict implementation method, while the improvement step requires flexibility.

The initial solution is divided into 3 main methods: SIH on clustered chains, SIH, Constructive ant. These 3 methods can be executed in parallel without communication and speed problems.

The improvement step depends on the queen to control procedures. Part of the purpose of the queen is to decide the granularity, i.e. number of processes, through feedback from the processes running. As discussed in the previous paragraph, communication is done on 2 levels.

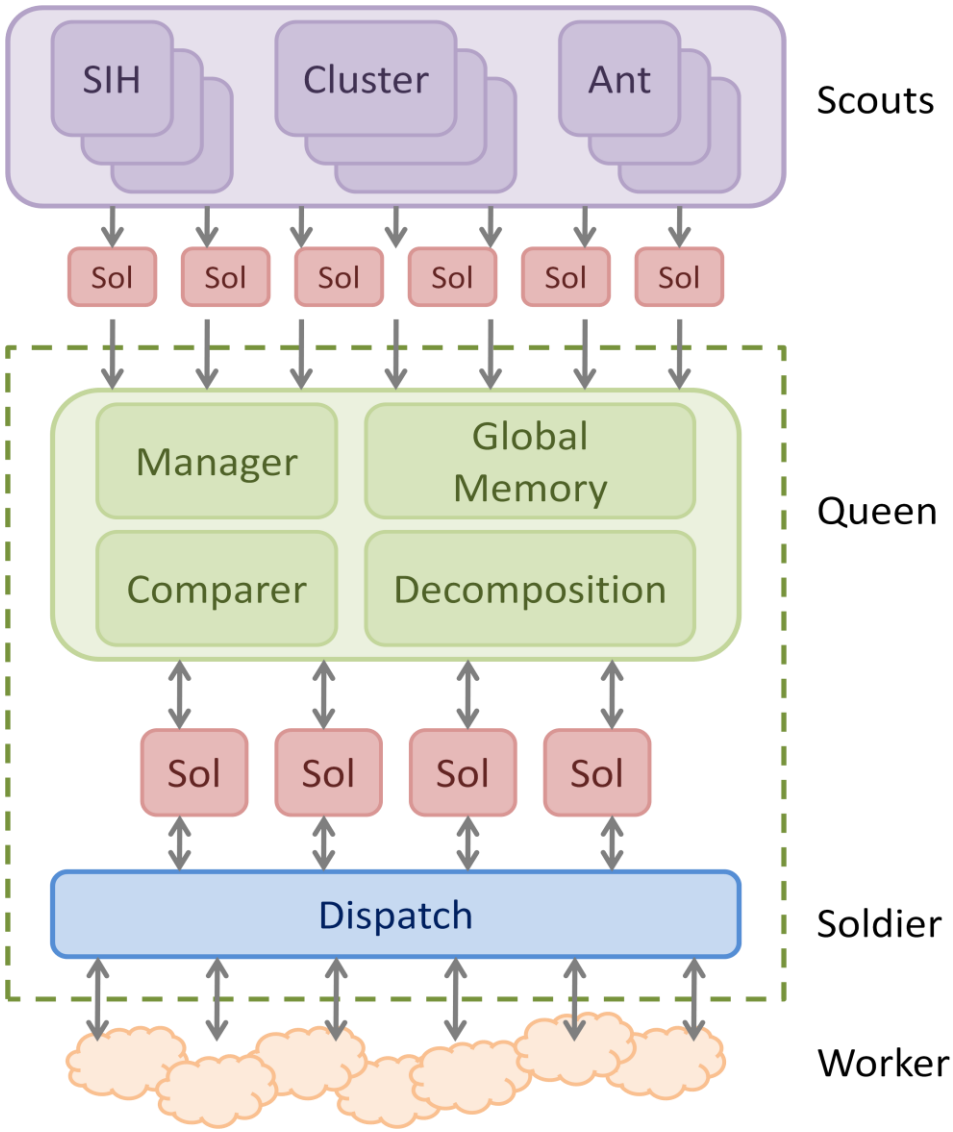

Figure 49: Organization of parallel search

The computational environment is not distributed computing where computers are connected with lower speed links. It is thus fine grain, closely connected architecture and the algorithms have been designed accordingly. The decomposition step, as illustrated in Figure 49, has the capability to decide on the number of solutions to use, thus determining the granularity of the system during run-time. Although the soldier decomposes the solution in more solutions, the parallel implementation will not be applied on that level.

6.6 Mapping

The mapping stage specifies where each task is to execute. In the initial solution stage, all tasks are viewed as equally important and are mapped to the same importance for process execution.

In the improvement stage, a sophisticated technique can be used. We define the following simple starting point of process priority:

1. Lowest cost – the solution with the lowest cost gets the primary consideration. The more complete initial solution method will result in a good solution for the majority of the time. The aim of this study to provide a feasible approach for business implementation, has a requirement that the algorithm to provide a solution as quick as possible. This priority will also set a realistic benchmark for other decisions.
2. Other – the rest of the solutions is next.
3. Scouts – it is important to keep scouting the area for uncharted territory. This step is only reached after a number of iterations, which indicates the complexity of the problem space. In this instance, the business will allow more iteration for a good solution.

The parallelization of the algorithm has many aspects to consider. The complexity of the unknown domain environment is an additional aspect to consider.

6.7 Parallel Ant System on Adaptive Objects

The previous paragraphs describe the steps to approach the parallelization of an algorithm. The partitioning paragraph recommended that the study focus on the initial solution and improvements stage for parallelization importance. This paragraph portrays the changes on the ASAO algorithm introduced in chapter 5.

The initial solution adaption is kept simple and can be interpreted in Figure 50. The algorithm spawns the initial solution methods in parallel. The expansion with new initial methods does not require a rework of the algorithm. Speed improvement on this fixed time part of the algorithm is notable and provides the option to reuse for diversification.

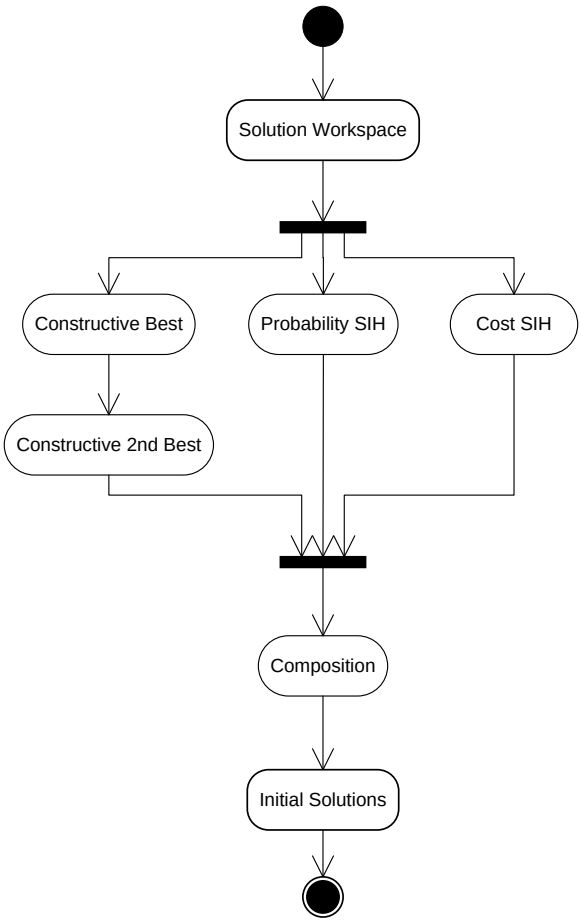

Figure 50: Initial PASAO

The result from the initial solution is evaluated and clearly marked to trace the origin for future references. A consolidated set of solutions is then passed on for further improvement. The two level control structures require partitioning of same type solutions to form the second level.

Figure 51 represents a high-level sequence diagram which indicates the purpose of each part of the system. It is important to note that once the algorithm has started, the queen spawns the process to the soldier. This entity continues to work until aborted by the queen. Figure 52 represents a more sequential flow of activities. From the diagrams, we can learn that the queen has a solution available at any time whilst the system is still executing. This approach provides flexibility which results in no adaption for each individual requirement. Best effort relates to the time allowed to execute.

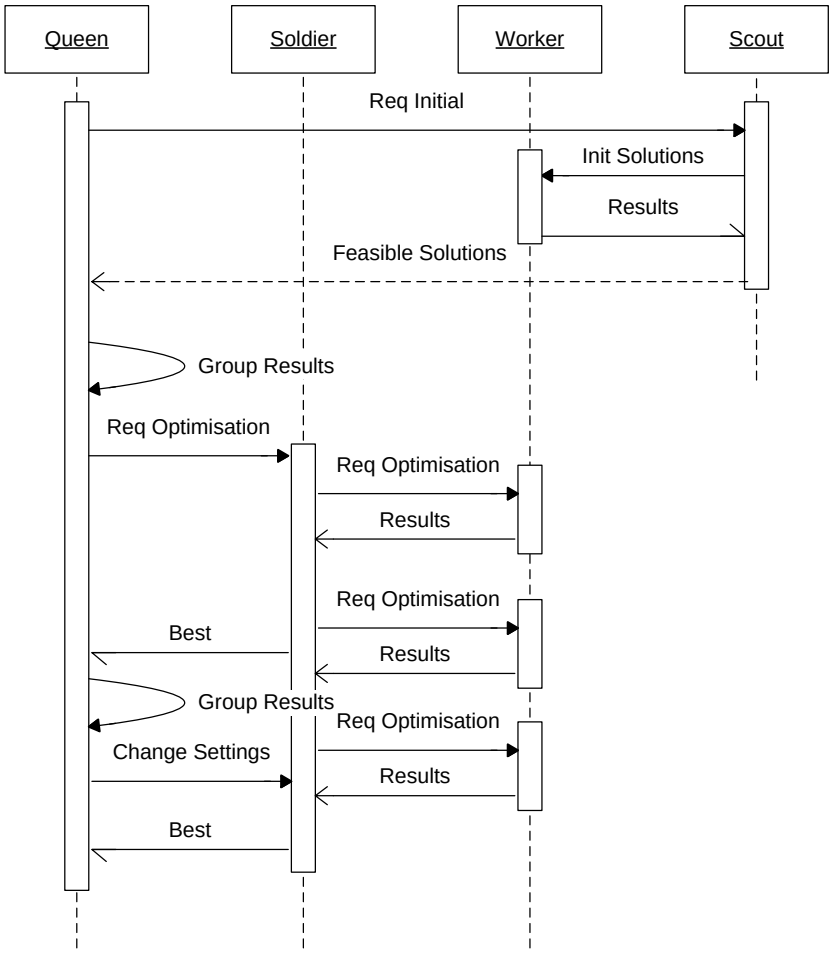

Figure 51: Sequence diagram for PASAO

The granularity of the solution is not known during design time for the improvement stage. The related diagrams must be interpreted accordingly. The same memory structure is used by the manager and the system. Each instance maintains its own structure's content.

Figure 51 display the call-back from the soldier to the queen as asynchronous. This is not clear in Figure 52, but can be interpreted that the 'join' action between the systems are not a dependency between the systems, but rather depends on the manager (queen). A 'join' command in C# force the main thread to wait until the thread has executed.

The straightforward transpose from the single threaded environment can be done in this way. This way, only utilization of multi-processor environment is achieved. The goal is to

simultaneously monitor the efficiency of the systems (soldiers) to be able to terminate where possible. Figure 52 must be interpreted with the asynchronous message return as described in Figure 51.

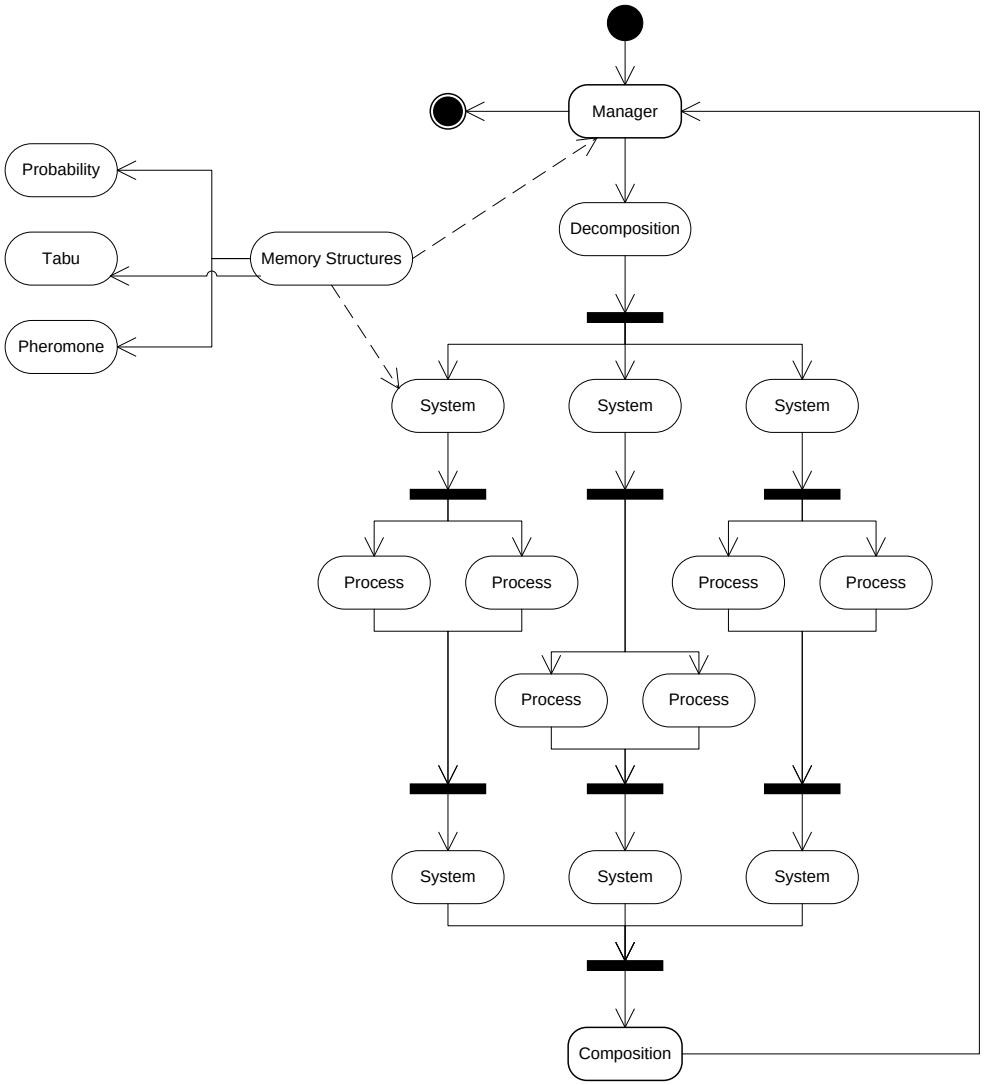

Figure 52: PASAO

6.8 Results

The primary goal for the parallel implementation is to utilize the multi-processor architect that is even available in today's personal computers. The solution results are expected to be the same or better as in a single threaded environment.

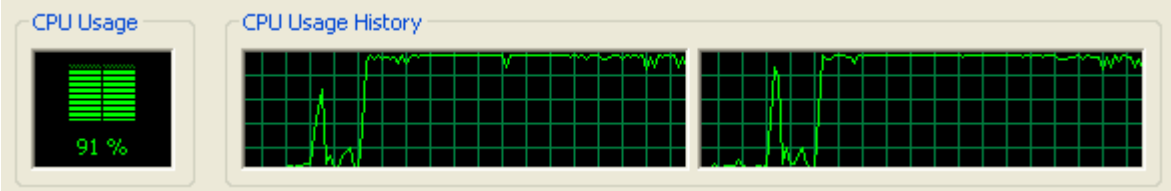

Figure 53: Desktop CPU usage multi-thread

The CPU processing depicted in Figure 53 clearly shows the improvement from the single threaded usage illustrated in Figure 47. It is also apparent where the initial solutions were generated, before the improvement started. The initial solution use a fixed time.

6.9 Summary

The parallelization of the ASAO algorithm has many possible benefits. This chapter explores some approaches. It designs some specific, not too complex, options for implementation. The advantage of utilizing multi-processes has immediate impact and cannot be seen as optional anymore, because of the availability of dual processor architecture in the basic computers on the market today.

Communication complexity between processes governs the design of the rest of the parallelization approach, i.e. partitioning, granularity and mapping. Combined with the adaptive object ambition of this study, this design should steer away from dependence on the domain knowledge. It was achieved by considering only what we know about the algorithm. As a result, each system contains the same memory structure and the queen has the decision on how to manipulate it. Inter process communication intervals are based on the Simulated Annealing methodology. The state of the cooling determines the communication required and the resulting actions to be taken.

Parallelization of VRP solutions is inevitable in today's implementation circumstances. The effect of implementation has various positive effects and can be clearly seen in the results.

7 CONLUSION

7.1 Overview

Logistics management is that part of the supply chain which plans, implements and controls the efficient, effective forward and reverse flow and storage of goods, services and related information. There are many opportunities to reduce total cost in supply chains. The prerequisites to many categories of cost savings, including supply chain management, are simplification, which corresponds to the basic lesson of Industrial Engineering 101: "Simplify before automating or computerizing"(Anderson, 2009).

The annual State of Logistics Report in the USA issued by the Council of Supply Chain Management Professionals shows that businesses spent a record $1.4 trillion on logistics in 2007. That's equal to 10.1 percent of Gross Domestic Product. It's the first time since 2000 that figure has exceeded 10 percent. Not surprisingly, most of the increases were related to fuel(Shulz, 2008). Transportation costs now account for 6.2 percent of nominal GDP. But capacity is permanently leaving the trucking industry as firms exit the market place and sell their equipment, often in foreign markets. The closing of Jevic Transportation, the nation's 71st-largest trucking company, is evidence that some truckers simply do not have adequate business plans in this era.

Solving different kinds of the Vehicle Routing Problem is an important area of Operations Research. Achieving improvement of only a small percentage may result in large. The cost of implementing a solution requires analysts and developers who understand the specific problem domain for the company. This initial cost of implementation is still a major drawback for companies to take the step towards implementing a solution.

Current study on the VRP focuses mostly on the efficiency and effectiveness indicators because it does not consider the problem as part of an enterprise system. This study reviews the importance of the indicators. Integration, both within and among cooperating enterprises, now comes first and is most important, providing the highest value. Dynamic integration creates the ability for many enterprises to participate within IT solution. Agility, the ability to react quickly, comes second.

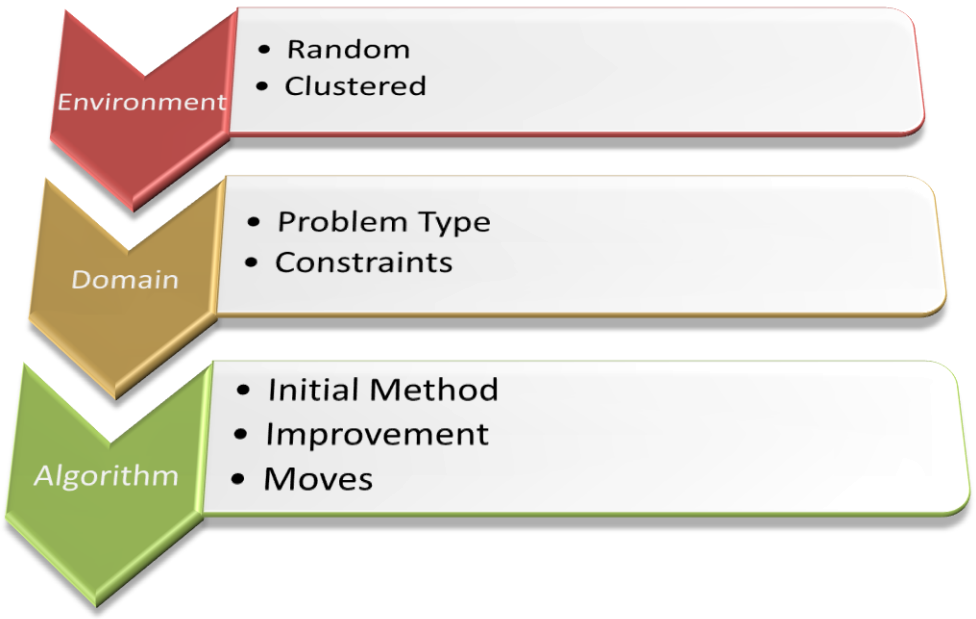

Figure 54: Traditional problem approach

This thesis addresses the problem of solving the VRP with variants through applying an adaptive solution which has no predefined knowledge of the problem environment. Current studies focus on solving VRP through identifying the type of problem beforehand and implement as solution for the specific type. Success is measured if an answer is *good enough* in a *reasonable time* for a problem where the user define the *constraint* and *cost* model.

7.2 Problem approach

The complexity of the VRP and all its variants has intrigued many OR researches. There exist numerous variants with numerous methods to solve them. The NP-hardness of the problem makes it then ideal playground for advance research. This study investigates the VRP problem with extraordinary requirements.

The innate ideas of solving the VRP consist of grouping stops according their geographical location. The expert will know that although the geographical location has a major impact on most real-life problems, the location is nothing else than a source for the cost calculation. The study seeks to steer away from the direct relation to the physical environment and view the

possibilities presented by the mathematical model. The inclusion of adaptive objects reduces the size of the mathematical model. The best-case approach in the initial environment to setup a cost matrix, contributes to the reducing of computational possibilities through the immediate elimination of impossible links.

The use of geographical independent clustering methods permits us to utilize traditional 'cluster first route second' methodologies. In combination with the pheromone trial, the results of clustering do not dominate the convergence. The memory structure for move types ensures that adaptiveness is also transferred to the algorithm. It is similar to solving the VRP with more than one algorithm.

7.3 Results

Results clearly indicate the accomplishment of the study. It must be acknowledged that the writer had a disproportionate advantage. The knowledge of designing the adaptive cost and constraint objects is strongly related to the problem domain as well as the internal algorithm use. This point toward the speed of the algorithm and not the efficiency.

The new definition on what qualifies as a good initial solution as well as the multi-start approach provides good coverage on the start of the algorithm. The addition of environment analysis influences the 'cooling' tempo of the convergence.

7.4 Summary

The design of a selective parallel heuristic algorithm for the Vehicle Routing Problem on an adaptive object model has been accomplish through integration of multiple methods. We conclude by stating that such a target would be impossible to achieve without hybrid methods. Adding the flexibility of determining run-time where and when to emphasize specific methods, contribute to the success of the design.

The study instigate numerous areas for further in-depth research of which the adaptive object model interface with the algorithm and the parallelization of the algorithm stands out the most. Both areas contain dependency on the memory structure used by the algorithm, as well as feedback from actions within the algorithm.

This study introduced an environment analysis model that has been dealt with from the static VRP problem viewpoint. The algorithm detects the type of environment during run-time. Further research on this method can assist in guiding the algorithm during the initial stage, which has enormous impact on the optimisation phase. A further development can include the re-evaluation of the environment when the optimisation memory structures have been set after a certain number of iterations.

A more complex problem such as the IRP (Inventory Routing Problem) can be implemented into the framework. The inventory routing problem involves the integration and coordination of two components of the logistics value chain: inventory management and vehicle routing. Inventory is required to provide cover against variability in demand, supply and the movement of products. Inventory is generally present in supply chain to ensure product availability. The inventory replenishment requirement is dependent on the time of visit, which is determined by the VRP.

A framework is a basic conceptual structure used to solve or address complex issues. A software framework is an abstraction in which common code providing generic functionality can be selectively overridden or specialized by user code providing specific functionality. This study introduces a framework for solving various types of complex VRP problems. A more in-depth study can be done on the differences between a framework base solution and a direct solution approach.

The parallel design discussed in the previous chapter is only the beginning of an important field of study for the VRP. There exist numerous approaches for parallel implementation from a computer scientist view. The multi-level usage of memory structures can be further expanded to fit the parallel approach. Problem granularity consists of the decision to breakdown the problem in smaller problems that can each be solved on its own and in parallel. Dynamic decision making on the feasibility of a granular level adds to the challenge and possibilities of parallel algorithms.

The methods utilised for this study can be applied on other problems and is not exclusive to the VRP only. The VRP provide a complex problem with multiple combinations that serves as a high-quality environment for research of new methods. This study steps towards an enterprise view for the VRP without sacrificing quality of the solution. This new approach opens up new possibilities.