A Multi-Agent Algorithm for Vehicle Routing Problem with Time Window

Hon Wai Leong and Ming Liu
Department of Computer Science
National University of Singapore
3 Science Drive 2, Singapore 117543
leonghw@comp.nus.edu.sg

ABSTRACT

Many existing algorithms for solving the vehicle routing problem with time windows (VRPTW) first construct initial tours and then apply a tour optimization algorithm to refine the solution. In this two-stage approach, the tour optimization stage is often hampered by the tour construction phase that produce initial solutions that are skewed, namely, the initial tours are very good, but the later tours are often very poor. This often leads to difficulties in the tour-optimization stage that often get trapped in local optimal quickly. In this paper, we propose a new multi-agent algorithm for solving the VRPTW that involves the uses a distributed, multi agent approach for the tour-optimization phase. Our approach can be considered as a combination of multi-agent system and heuristic local search. A prototype system has been developed and extensive experimentation on the Solomon benchmarks show that our multi-agent approach is effective and has comparable performance to the best results in the literature.

Categories and Subject Descriptors

I.2.11 Distributed Artificial Intelligence – *multiagent systems*

Keywords

Multi-agent algorithm; vehicle routing problem (VRPTW).

1. INTRODUCTION

The *Vehicle Routing Problem with Time Window* (VRPTW) problem deals with the problem of serving a set of customers with given service time windows using a fleet of capacitated vehicles in order to minimize the service cost by minimizing the number of routes (number of vehicles) and minimizing the total travel distance. The VRPTW problem is encountered in many industries, such as fast-food deliveries, product deliveries, scheduling of service appointments. Optimizing the solutions may lead to lower operational cost, greater profit, and better service to customers. Thus, VRPTW problem is of growing interest in management science, logistics management and computer science.

In a VRPTW instance, we have three types of entities: customers, routes (vehicles), and a central depot. A depot has a deadline for vehicle to return. A vehicle, which serves one route, has a fixed *traveling speed* and *capacity*. The sum of the *demands* of customers in the route cannot exceed the vehicle capacity. A customer has its own *demand*, *service time*, and a *time window* for vehicle to serve. If a vehicle arrives at customer before the starting time of time window, it has to wait until the starting time to serve the customer. Vehicle cannot serve customer if it arrives after ending time. Each customer can only join one route (vehicle). The objectives of solving VRPTW (in priority order) are: *minimize the number of vehicles* (*routes*) and *minimize the total travel distance of all the vehicles.*

The VRPTW has been extensively researched. A standard two-stage solution approach is commonly used, namely, a *tour-construction stage* and then, a *tour-optimization stage*. For the tour-construction stage, a popular "greedy" algorithm is the *Push Forward Insertion Heuristic* (PFIH) by Solomon [7]. The PFIH algorithm considers all customers which can be feasibly inserted into the current tour. When there is no more feasible customer to insert into current tour, PFIH starts a new route. Therefore, the remaining customers are increasingly difficult to insert. This brings a problem of creating skewed solutions, i.e. the routes built at the earlier stage have quite a number of customers, while the routes built at the later stage contain very few customers. For tour-optimization stage, heuristic like Exchange Heuristic by Potvin and Rousseau [5], Tabu Search by Taillard *et al* [8], Genetic Algorithm by Thangiah [9] and Simulated Annealing by Chiang and Russel, [1]. Gambardella *et al* proposed a Multiple Ant Colony System to tackle VRPTW [3]. All these heuristics attempt to solve the problem of getting trapped in local optimal.

In this paper, we present a novel algorithm that adopts a multi-agent approach in the tour-optimization stage in order to tackle the problem of being trapped easily in the local optimal. Our Multi-Agents approach treats each of the customers, routes, and the global planner as an agent and allows them to exchange information and respond to the environment. Our algorithm uses a push forward insertion heuristic (PFIH) [7] to obtain an initial tour. After that, our multi-agent algorithm takes over to optimize the tour and the focus of our algorithm is on this multi-agent approach to tour optimization. Each agent has a set of operations that are aimed at improving its own local objective while moving the global solution state towards better global solutions. In addition, the planner agent also periodically performs global moves that are aims at improving the global solution state. Our current implementation of the multi-agent algorithm for the VRPTW is still not very matured. However, our experiments show that the multi-agent algorithm is able to obtain solutions that are competitive with the best solutions for the well-known Solomon benchmark datasets.

SAC'06, April, 23-27, 2006, Dijon, France.

The rest of the paper is organized as follows: Section 2 describe the PFIH construction of the initial tour. Section 3 gives an overview of our multi-agent approach and then presents the detailed design for our *Multi-Agent Algorithm* (MAA) for the VRPTW. In Section 4, we evaluate the performance of our MAA and give comparison results for the Solomon benchmarks. Section 5 summarizes the contributions of this paper.

2. PFIH for Tour Construction

We use Push Forward Insertion Heuristic (PFIH) [7] to build routes in the tour-construction stage. The PFIH repeatedly inserts the "best feasible" customer into the current partially built route. The "best feasible" customer is defined to be the one with the least cost computed by a cost function. If there is no more customers with feasible insertions can be found, it starts a new route unless all customers have been routed. When inserting a customer *u* into a route *R*, there is a *push forward* in the time to begin service at all the customers after *u* in the route *R*. Therefore, we call it Push Forward Insertion. Because of its greedy nature, it becomes increasing hard to "pack" un-routed customers into the later routes. We observed this phenomenon after running PFIH on the Solomon Benchmark Data Set. The first several routes contain quite a number of customers, while the last several routes contain only small number of customers. This is contrary to the objective of minimizing the number of routes.

3. OUR MULTI-AGENT ALGORITHM

A *M*ulti-*A*gent *S*ystem (*MAS*) is a system in which many intelligent autonomous agents share a common environment and interact with one and another ([2], [6]). Upon perceiving and sensing the environment, each agent responds by performing certain actions. Although, agents are aware of their environment, its perception is normally localized. Agents share a common goal. However, they can pursue their own interests as well. No single agent can understand the whole problem, and those agents must work together to solve the problem and reach the objective.

Multi-Agent systems are widely used in various applications such as air traffic, process control, manufacturing, supply-chain management, robotic control, and games, etc. This is due to the fact that MAS presents a very intuitive approach in understanding, designing and implementing distributed and concurrent software.

3.1 Why a Multi-Agent Approach

The purpose of adopting a multi-agent approach in tour-optimization stage is to create more opportunities for the solution to jump out of local optima, and increase the probability of reaching global optimal. Instead of using a single, centralized program to control the optimization procedure, we distribute the controls to all of the entities in our multi-agent system. The multi-agent approach has been used to solve Inventory Routing Problem by Lao and Leong [4]. Our system can be considered as a combination of multi-agent system and heuristic local search.

The multi-agent approach gives a natural and intuitive way for designing and implementing the software system for solving the VRPTW problem. Each entity in the VRPTW instance is treated as an agent and the VRPTW instance is represented by a group of interacting agents. Each agent has its own local objective. The agents can perform a set of operations/moves to change its status by moving to its individual objective and affect the environment and modify other agents' status. Besides operations of achieving the local objective, each agent is also responsible for satisfying constraints. Each agent's local objective partially represents the common system objective. Thus, while agents pursue their local objective, the overall system is moving towards the common objective.

We predict that with the diversity that Multi-Agent system provides and the general moves towards approaching common objective that is generated by agents, we will be able to search solutions in a larger neighborhood and increase the probability to reach global optimal.

3.2 Overview of Our Multi-Agent Algorithm

In a real life VRPTW, *customers* can exchange information with one another. Each *route* (or vehicle) knows the information on the customers in its route and exchange information with other routes. A *global planner* (symbolized here by the depot) is in charge of all the routes and knows all the information of customers it is going to serve. Each of these entities has their own objective. The objective of each customer is to *get the service as soon as possible* and to minimize its waiting time. The objective of each route objective is to *maximize the usage of its capacity* and *minimize the traveling distance*. The objectives of the customers and routes are often in competition with each other. Therefore, there is a need for the *global planner* who has the overall picture of the current state and is able to *coordinate the operations* of the customers and the routes. The objective of the global planner's objective is to *minimize the number of routes* (*vehicles*), and *the total distance traveled* in all the routes.

In our Multi-Agent Algorithm for the VRPTW (MAA-VRPTW), we model each customer by a *customer agent*, each route by a *route agent*, and the global planner by a meta-agent called the *planner agent*. Each customer agent knows its current "state", namely, the route that it belongs to, and when it is scheduled to be serviced. The state of each route agent is given by *the set of customers in its route*, as well as the *sequence of the customers along the route*. The "solution state" is represented by the composite state of all the customer agents, the route agents. The planner agent knows the global solution state and its objective is to *minimize the number of routes* (*vehicles*), and *the total distance traveled* in all the routes.

Each customer and route agent has its own local objective and a set of operations that are use to modify its current state to achieve its local objective. The operations of the customer and route agents tend to be "local" operations based on its local view of the current "solution state" and they are often in competition with each other. The planner agent is responsible for *coordinating the competing operations* of the customers and the routes. To do this, the planner agent maintains an "*move pool*", which consists of the "best moves" proposed by the customer and route agents, one "best move" per agent. The planner agent then examines the moves in the move pool and selects the move to be executed next. The selected move is then performed and the solution state is then modified (locally).

In addition, the planner agent periodically activates "global" operations that attempts to reduce the number of routes by trying

to remove "bad" routes (as described later) and to optimize individual routes (by reordering the customers in some routes using methods such as 2-opt.)

Generally, our MAA-VRPTW works in the following way (as illustrated in Figure 1): First of all, it initializes data for agents (customer agents, route agents, and planner agent) using the solution obtained from the PFIH initial route construction. It then activates all the agents and the initial move pool. After that the multi-agent system takes over as agents iteratively propose moves, the planner agent coordinates the selected move, and periodically makes global moves, all according the environment they individually sense. Finally, solution is written into the file system.

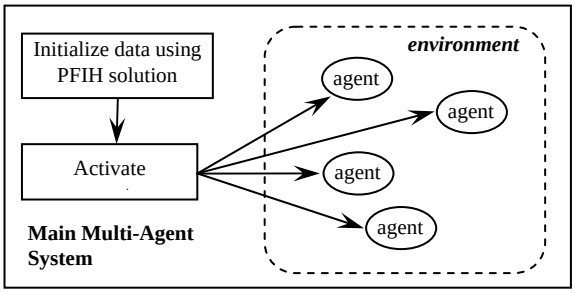

Figure 1. Flow Chart of Our Multi-Agent System

3.3 The Cost Function

We define a cost function in multi-agent system to evaluate the goodness of the agents' moves as well as the goodness of solution

$$Cost = \lambda * N + D \quad (1)$$

where *Cost* is the cost of this solution, *N* is the number of routes in the solution, and *D* is the total travel distance by all routes. The parameter λ, is a constant, which increases the relative importance of the number of routes. Since the primary objective in VRPTW is the number of routes, we usually set λ to be a relatively large number. This implies that decrement in the number of routes causes a large reduction in the cost. For the rest of this paper, we use the term *cost* to refer to the cost computed by this function.

3.4 Agent Operations

We should design operations that drive agents towards their own local objectives and also advance the system state to those that achieve the system's common objective. An agent's objective should be consistent with the system's common objective, and it should be restricted in a local scope. Figure 2 shows the interaction among customer agents, route agents and the planner agent. The planner agent takes the role of overall controller of the whole multi-agent system. Planner agent, Customer Agents and Route Agents can communicate with each other. The agents' behavior of sensing the environment, communicating with other agents, acting based on the perception is transformed into several operations in the Multi-Agent system.

Types of Agent Operations: There are two types of agent operations: *active moves*, and *passive moves*. An *active move* is a move that originates from the agents themselves and is usually aimed at optimizing the agent's own local objective (such as reducing the traveling distance for a route agent). These moves are proposed by the agents and included into the move pool and if selected, are performed by the agents themselves. In contrast, passive moves are operations that an agent must carry out on behalf of other agents (usually the planner agent) or upon request from other agents. Passive operations can also be considered to be internal methods of the agents that are activated by other agents. Usually, passive moves are aimed at reducing number of routes. Route agents' passive move is also used by customer agents. Passive moves of route agents and customer agents are controlled by planner agent. This means that planner agent gives order to route agents and customer agents to perform operations.

***Operations of the Customer Agent*:**

Each customer agent represents one customer in the VRPTW. Customer agent contains the customer's attributes of position (x, y-coordinates), service time, time window, and demand. Besides these attributes, customer agent also records the route to which the customer belongs. The objective of customer agent is to be serviced by a route as soon as possible in its service time window. Based on this objective, operations for customer agent are as follows. (In each of the operations below, we let c_0 be the customer represented by the customer agent and assume that it belongs to the route r_0.)

C-BI – Customer-Best-Insert Operation (*Active*): In this (active) operation, the customer agent proposes the "*best feasible location*" to insert itself (customer c_0). When customer c_0 leaves the current route, r_0, and joins another route, say r_1, there is a change in the cost. The "best feasible location" is the location in

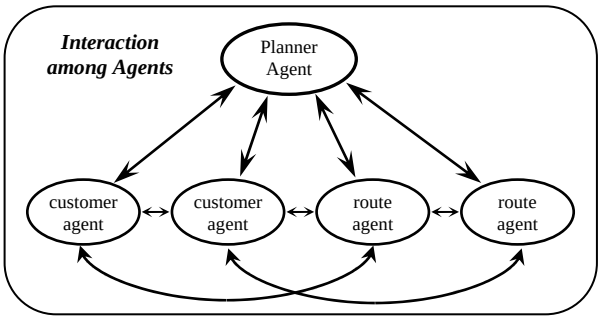

Figure 2. Interaction among Agents

another route that is feasible to insert customer c_0 and that will give rise to the *greatest cost reduction* (or smallest cost increment if there is no cost reduction). To compute this, the customer agent considers each route and calls the best-insert operation (to be explained later) for that route and choose the best one.

C-BE – Customer-Best-Exchange Operation (*Active*): In this (active) operation, the customer agent proposes the "*best customer exchange*" between customer c_0 and another customer c_1. The "*best customer exchange*" is defined to be the exchange with the greatest cost reduction (or smallest cost increment if there is no cost reduction). This operation is an active move and feasibility checking is performed.

C-PR – Customer-Propose-Route (*Passive*): This is a passive operation (usually called by the Planner Agent) that proposes a route that the customer, c_0, would like to join so as to reduce the waiting time of service. In accordance with its local objective, the customer agent proposes route from the nearest to the furthest to it. The distance from customer to a route is defined as the Euclidean distance from customer to the centroid of the route.

This operation belongs to passive move and it does not perform feasibility check (which is left to the planner agent).

C-PCR – Propose Customer Replacement (*Passive*): This is a passive operation (usually called by the Planner Agent) that proposes a customer (from another route) to replace itself in route r_0. in order to reduce the waiting time of service as well as the cost. Again, the customer agent proposes customer from the nearest to the furthest neighbor to itself and does not perform feasibility check.

Operations of the Route Agent:

Each route agent represents one route served by a vehicle and the information it maintains includes (i) information on the vehicle (vehicle capacity, speed) and (ii) information on the route – the set of customers assigned to this route, the order of customers along the route, total travel time, travel distance, total demand, the centroid of this route, and the status of this route. The status indicates whether this route is a "bad" route (to be defined later). The objective of the route agent is to maximize its capacity utilization and minimize the travel distance. Based on the objective, the operations of route agents are defined as follows. (In the operations below, we let r_0 be the route represented by the route agent.)

R-BI – Best Insert (*Active*): In this (active) operation, the route agent proposes the "*best customer"*, say c_1, to insert into itself (route r_0). When the customer c_1 leaves the other route to join route r_0, there is a change in the cost. The "best customer" is defined to be the customer from another route that can be feasiblly inserted into the current route r_0 and that will give rise to the *greatest cost reduction* (or smallest cost increment if there is no cost reduction).

Note: This operation is similar to the customer agent's best insert operation, C-BI – in that both operations search for a "best insert" for a customer *c* into a route *r*. However, they are different – in C-BI, the customer *c* is fixed, while the route *r* is varied. In R-BI, the route *r* is fixed, while the customer *c* is varied.

Customer agent checks with route agent where the best insertion place is using this operation.

R-BE – Best Exchange (*Active*): This is an active operation that proposes the "*best customer-customer exchange*" between a customer in route r_0 and a customer from another route. "Best" exchange is the exchange with the greatest cost reduction (or smallest cost increment if there is no cost reduction). This is an active move. (There are three types of best exchange operation in increasing computational complexity. The first is to check against *a specific customer* in another route, another is to check against *all the customers in another route*. And the third type is to check against *all other customers in other routes*.)

R-IC – Insert Customer (*Passive*): This is a passive operation (usually invoked by the Planner Agent) that inserts a customer into itself (route r_0). This operation is usually done to maximize the capacity utilization of the route. (An example is when the planner agent attempts to "destroy" a bad route by inserting all the customers into other routes.)

R-RC – Replace Customer (*Passive*): This passive operation (usually invoked by the Planner Agent) tries to replace a specified customer c_1 in its route r_0 with another specified customer c_2. Feasibility check is performed for this operation. Usually, the objective of this operation is to try to create more opportunity for a subsequent insert customer operation to be successful. Note that, in some cases, c_1 (the replaced customer) may, in turn, try to replace other customers.

Operations of the Planner agent:

The planner agent plays the role of the global planner in the VRPTW and it has information on all the customer agents and route agents. It has the responsibility of coordinating the often competing moves (active operations) proposed by the customer and route agents into the move pool. The planner agent selects the move to be performed at each iteration of the multi-agent algorithm. In addition, the planner agent has the overall objective of minimizing the *number of routes* (primary objective) and the total travel distance of the routes (secondary objective). To reach this overall objective, the planner agent periodically performs two global operations, namely, (*intra-*) *route optimization* and *bad route removal*. These operations are described further below.

P-BMS – Best-Move-Selection: This is the "default" operation of the planner agent – namely to coordinate the moves in the move pool that are proposed by the customer and route agents and to select the best move to be performed. This is set up as follows: after the agents are initialized, each customer agent, based on its knowledge of the current environment (solution state), proposes moves using its active operations – *Customer-Best-Insert* (C-BI) and *Customer-Best-Exchange*, (C-BE). Similarly, each route agent also proposes moves based on its active operations – *Route-Best-Insert* (R-BI) and *Route-Best-Exchange* (R-BE). These proposed moves are put into the move pool. All the proposed moves in the move pool are feasible with respect to the current environment since feasibility check has been performed for all active operations. The planner agent then selects the best move from the move pool. The operation that corresponds to the best move is performed, and the corresponding agent can re-propose a new move of the same operation type to the move pool.

After each move, the environment has changed slightly and this may affect some of the moves that has been proposed in the move pool. Theoretically, we can resolve this by first updating the state of all the agents and asking them to re-propose feasible moves to the move pool. However, this incurs a very high computational cost on the multi-agent system. In our current implementation, we adopt the following compromise – the state of the agents that are changed are updated after each move. However, the agents do not re-propose feasible moves to the move pool after each move is performed. This means that the information in the move pool may be slightly out-of-date and a small number of the moves in it will become infeasible and there is a need to check the feasibility of the selected move. However, we do not need to constantly check all the moves in the move pool, but only those that are selected as the best moves. Periodically, the move pools is re-initialized by emptying it and asking all the agents to re-proposed moves into the move pool based on the up-to-date environment at that time. (We note that a more sophisticated mechanism can be used to "trigger" the renewal of the move pool – however, this has not been incorporated in our current prototype.)

P-RO – Route-Optimization: This is a *global operations* invoke by the planner agent to reorder the customers in a route using well-known heuristics algorithms to reduce the distance traveled

of the route. The set of customers in the route is unchanged. This sub-problem is an instance of the well-known traveling salesman problem and we use the 2-opt exchange heuristics to reorder the customers in the route. This global operation is needed because after a number of local moves by the customer and route agent, the ordering of the customers within a route may be far from optimal and this may make it *unnecessarily expensive* or impossible for new customers to be inserted into the route. After reordering, it may be "easier" for new customers to be inserted into the route.

P-BRR – Bad-Route-Removal: This is a *global operations* invoke by the planner agent to directly attempt to reduce the number of routes by removing "bad" routes. As was describe earlier, it is often observed in many VRPTW solutions that there is a mixture of good routes and bad routes in a given solution. To help define a bad route, we first define a "goodness measure" for each route r, that is given by $\boldsymbol{G(r) = nc(r) + uc(r)}$, where $nc(r)$ is the number of customers in the route r, and $uc(r)$ is *used capacity* (sum of the customer demands) of the route r. We define a threshold G_0 given by $G_0 = \alpha \cdot nc_{max} + \beta \cdot Ca(r)$, where nc_{max} is the maximum number of customers among all routes, $Ca(r)$ is the capacity of the route (vehicle) r, and α and β are constants. (For our current implementations, we use $\alpha = 0.5$ and $\beta = 0.75$ which means that if the number of customers in a route is below *half* of the maximum number and the total demand is below 75% of the capacity, the route is considered a "*bad*" route.

The planner agent tries to "remove a bad route" by transferring the customers in the bad route to other routes, wherever possible, by making use of the passive operations of the customer and route agents. If this attempt successfully removes all the customers in the bad route, then the bad route is removed and the number of routes is reduced by one.

The rough process is as described as follows: The bad routes are considered in decreasing order of their goodness measure G(r). For a given bad route r, and customer c in the route, the planner agent first use sub-operation *find-insert*(c) to try to insert customer c into another route. If that is not successful, (namely, it is not possible to insert customer c into *any* other route), the planner agent next calls *find-replace*(c) that tries to have customer c replace some customer, say c_1, in another route r_1 that can be either inserted into another route. (More details can be found in (Leong and Liu, 2005.)

Implementation of our MAA-VRPTW: We have implemented our multi-agent algorithm for solving the VRPTW, called MAA-VRPTW. However, this is not implemented using a *distributed* multi-agent system. Instead, the multi-agent system is simulated with a discrete event simulator. This helps to greatly simplify the implementation since there is no need to implement sophisticated agent communication and negotiation process and protocols. For our purpose of studying the effectiveness of the multi-agent approach in solving an optimization problem VRPTW, this simulation method suffices. Our implementation is done in C++.

4. EVALUATION

We use the well-known Solomon Benchmark datasets to evaluate the performance of our multi-agent solution approach for the VRPTW. The Solomon Benchmark contains 56 datasets, each of which has 100 customers. The characteristics of the datasets are shown in the Table 1 where "C" means clustered, "R" denotes random and "RC" means random-clustered.

Test Data Set	Time Window	Num. of Routes*
C101 – C109	Tight	10
C201 – C208	Loose	3
R101 – R112	Tight	9-18
R201 – R211	Loose	2-4
RC101 – RC108	Tight	10-14
RC201 – RC208	Loose	3-4

Table 1. The Solomon benchmark dataset. The letters "C" (clustered), "R" (random), and "RC" (random-clustered) refer to distribution of the cities. The last column indicates the best known "number of routes".

We believe that Multi-Agent System moves the solution around the neighborhood and thus makes it jump out the local optimal to approach global optimal. This is illustrated in Figure 3, which shows a plot of the solution cost function over time for the dataset R208. The y-axis is the cost function, $Cost = \lambda * N + D$, of the solution. The horizontal axis is the number of iterations – each iteration is the time interval between renewals of the move pool. The upper line (in blue) shows the change in the cost function over time while the line at the bottom (in pink) is the best known solution cost for dataset R208. It is clearly shown that there are small changes in the solution for the first five iterations. This means the system is making moves that enhance the chance of reducing the number of routes, which occurs around the 6th iteration. After the 6th iteration, the solution gradually approaches

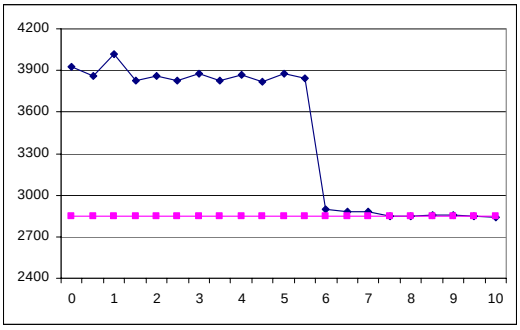

Figure 3. Plot of the Cost Function for Dataset R208. (The bottom line is the cost of the best known solution.)

the best known solution for R208.

4.1 Comparision with Best-Known Solution

We first compare the results obtained by MAA-VRPTW with the best-known solutions on the 56 Solomon benchmark. The objectives of our MAA-VRPTW are the same as that of other algorithms for the VRPTW, namely, in priority: minimize the number of routes first, and then minimize the total travel distance. The results are summarized in Table 2.

Type	#Datasets	#(N=N*)	#(D ≤ D*)	#(D ≤ 1.05D*)
C1	9	9	2	6
C2	8	8	4	2
R1	12	3	0	3
R2	11	8	0	1
RC1	8	0	0	0
RC2	8	6	0	0
Total	**56**	**34 / 56**	**6**	**12**

Table 2. Performance of MAA-VRPTW against the best known solutions (using the number of datasets that achieve the *minimum number of routes* N*, *minimum total distance* D*, and total distance within 5% of D*).

From Table 2, we see that our MAA-VRPTW solves 34 (out of 56) datasets with the same number of routes as that of the best known solution. Among these 34 dataset solutions, six of them has the same total distance traveled, is equal to or even slightly better than best known solutions, and 12 have the total travel distance exceed the best known solutions by less than 5%.

4.2 Comparision with Other Algorithms

Finally, in Table 3, we show a performance comparison of our MAA-VRPTW with a number of other algorithms in the literature for solving the VRPTW. These methods includes MACS-VRPTW (Gambardella et al, 1999) which uses the ant colony system, GEDION (Thangiah, 1993) that uses genetic algorithm, SA (Chiang and Robert, 1996) that uses simulated annealing, and TB (Taillard et al, 1997) that uses tabu search. For each entry, the first number is the number of routes, the second is the total distance traveled, while the third number is the CPU time in seconds. For MAA-VRPTW, the CPU time is on a PIII 1 GHz machine running Linux. The results for the other methods are based on those reported in their respective papers.

	MAA-VRPTW	MACS-VRPTW	GIDEON	SA	TB
C1	***r=10*** ***d=839.4***	10 828.40	10 892	10 930.8	10 830.41
C2	***3*** ***622.33***	3 593.19	3 749	3 666.3	3 592.75
R1	***12.92*** ***1230.92***	12.55 1214.80	12.8 1300	12.58 1300.4	12.64 1233.88
R2	***3.09*** ***1003.71***	3.05 971.97	3.2 1125	3 1163.1	3 1046.56
RC1	***13*** ***1402.04***	12.46 1395.47	12.5 1474	12.38 1474.1	12.08 1404.59
RC2	***3.62*** ***1214.92***	3.38 1191.87	3.4 1411	3.38 1393.7	3.38 1248.34

Table 3. Comparison of MAA-VRPTW with other algorithms (using the number of routes (1st number), and total distance (2nd number)).

For clustered data (C1 and C2), our MAA-VRPTW ranks 3rd among these five algorithms and it gives a good performance. While for random and random-clustered data, MAA-VRPTW also gives a reasonable performance. We can see that, with a relatively good performance, our MAA-VRPTW that uses the multi-agent approach solves the problems in a very short time. However, the computational times cannot be directly compared because different computers are used for these systems.

5. CONCLUSION

In this paper, we presented a multi-agent approach to solve Vehicle Routing Problem with Time Windows (VRPTW) which we call MAA-VRPTW. In this approach, each entity in the VRPTW (customer, route, and global planner) is represented by an agent. Agents can have their local objectives that are consistent with the global objectives in VRPTW, namely minimize the number of routes and then the total distance traveled. In our multi-agent solution to the VRPTW, each agent can perform a number of operations (that modifies the solutions) that optimizes its local objectives. The planner agent coordinates these moves by the customer and route agents and periodically performs global moves that may drastically change the solution state in its attempt to minimize the number of routes and total distance traveled. Our MAA-VRPTW first uses a push forward insertion heuristic (PFIH) to obtain an initial solution. After that, the multi-agent algorithm takes over to continuously refine the solution using the set of moves that are coordinated by the planner agent. We have implemented our MAA-VRPTW by simulating the execution of a distributed multi-agent system. Though our MAA-VRPTW is not very matured, it achieves good results that are comparable to those of well-studied methods in the literature on the Solomon benchmark datasets. The results suggest that the multi-agent approach shows great promise in solving the VPRTW and other similar optimization problems.

6. ACKNOWLEDGMENTS

This research is support in part by the National University of Singapore under grant R252-000-128-112. This work was carried out while the second author was an undergraduate at the National University of Singapore.

7. REFERENCES

[1] Chiang, W.C. and Russel, R.A. (1996), "Simulated Annealing Metaheurstics for the Vehicle Routing Problem with Time Windows", Annals of Operations Research, Vol. 63, 1996, pp. 3-27.

[2] Franklin, S. and Graesser, A. (1996), "Is it an Agent, or Just a program?: A Taxonomy for Autonomous Agents", Proceeding of the Third Intl Workshop on Agent Theories, Architectures, and Languages, Springer-Verlag, 1996.

[3] Gambardella, L.M, Taillard, E. and Agazzi, G. (1999), "MACS-VRPTW: A Multiple Ant Colony System for Vehicle Routing Problems with Time Windows", Technical Report IDSIA, IDSIA-06-99, Lugano, Switzerland, 1999.

[4] Lao, Y.Z. and Leong H.W. (2002), "A Multi-Agent Based Approach to the Inventory Routing Problem", Proceedings of PRICAI-2002, LNAI-2417, pp. 345-354.

[5] Potvin, J.Y. and Rousseau, J.M. (1995), "An Exchange Heuristic for Routing Problems with Time Windows", Journal of the Operational Research Society, Vol. 46, 1995, pp. 1433-1446.

[6] Schumacher, M. (2001), "Objective Coordination in Multi-Agent System Engineering: Design and Implementation", Berlin, New York: Springer, 2001.

[7] Solomon, M.M. (1987), "Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints", Operations Research, 35:2, 1987, pp. 254-264.

[8] Taillard, E., Badeau, P., Gendreau, M., Guertin, F. and Potvin, J.Y. (1997), "A Tabu Search Heuristic for the Vehicle Routing Problem with Soft Time Windows", Transportation Science, Vol. 31, No. 2, 1997, pp. 170-186.

[9] Thangiah, S.R. (1993) Vehicle Routing with Time Windows Using Genetic Algortithms. "Applications Handbook of Genetic Algortihms: New Frontiers", 1993.