APPLICATION OF EVOLUTIONARY ALGORITHM IN SUPPLY CHAIN MANAGEMENT

M.H. Lim and Y.L. Xu

School of Electrical and Electronic Engineering,
Nanyang Technological University,
Singapore 639798
(emhlim | pg04266316)@ntu.edu.sg

ABSTRACT

One important aspect of supply chain management is the maintenance of an efficient and cost-effective system for the distribution of goods or products to retailing outlets. Nowadays, it is almost inevitable that an efficient distribution network can only be achieved with the support of computer based optimization software. We show in this paper how an evolutionary algorithm (EA) can play an effective role in this aspect. Besides being an effective and efficient optimization tool capable of handling problems of significant computational complexity, it enjoys great flexibility in coping with the constraints of typical real-life supply chain problems. Our work considers the problem of managing a distribution network for replenishing the supply of fuel to refuelling stations located all over the island. It is designed to generate a set of routes for a fleet of fuel trucks. The trucks are dispatched according to a pre-specified plan to replenish the supply of the various refuelling stations subject to several dynamic constraints. Our solution methodology is an EA capable of generating approximate solutions with reasonable computational time.

Keywords: vehicle routing problem, combinatorial optimization, fuel truck dispatch system, evolutionary algorithm, travelling salesman problem.

1. INTRODUCTION

The applications of *vehicle routing problem* (VRP), which was first proposed by Dantzig and Ramser (Dantzig and Ramser, 1959) in 1959, are very common in real life. It can be described by the scenario that follows. Consider a depot having a fleet of vehicles with limited capacities and a set of customers, each with a certain demand for the merchandise or goods to be dispatched. The problem is to determine optimal routings for each vehicle to visit every customer exactly once in order to fulfil the demand. The most common goal for optimization is to minimize the overall distance travelled by the vehicles.

An illustration is as shown in Figure 1 where there are 3 vehicles and 8 customers or stations in the delivery network. The goal of optimization is to route all 3 vehicles such that all the 8 stations are visited by one of the vehicles. The plan for routing the 3 vehicles is such that the total distance travelled by the vehicles is minimal. Figure 1 presents an illustration of one possible plan of routing the three vehicles starting and ending at the depot. It should be emphasized that in reality, for many real-life problems, it is common for practical constraints or limitations to be considered in the process of optimization.

A common variation of the VRP can be derived by considering the limitation in the capacity of each vehicle. This is referred to as the *capacitated* VRP (CVRP). Another variation of the VRP considers the scenario where some or all the customers are to be supplied within a certain time window. This is referred to as the VRP with time windows (VRPTW). In minimizing the overall distance travelled, the derived solution also must not violate the specified constraints. It is clear that after clustering the customers according to the vehicle that services them, the routing of each vehicle is essentially the famous travelling salesman problem (TSP).

This is why the VRP is sometimes regarded as a generalization of TSP, and is therefore NP-hard. It is considered as one of the most difficult combinatorial optimization problems.

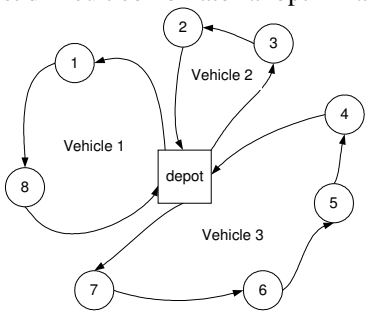

Figure 1. Illustration of a 3-vehicle routing problem.

Due to the practical relevance of VRP and its considerable difficulty, it is therefore not surprising that the VRP continues to be a subject of intense study in the past few decades. Many solution techniques have been proposed to solve the VRP. From literature survey, one can classify the methods into exact, heuristics and meta-heuristics approaches. For example, Fisher proposed a branch and bound algorithm, which is an exact enumeration approach for solving VRP (Fisher, 1994). Although it succeeded in solving a problem with 71 customers to optimality, it failed to solve some smaller instances. Due to the NP-hardness of VRP, this approach can only handle the instance with up to 100 customers. This is why heuristics and meta-heuristics were proposed for solving VRP with approximate solutions in reasonable time. Some examples include 2-phase algorithm (Fisher and Jaikumar, 1981), tabu search (Kelly and Xu, 1996; Taillard et. al., 1997; Gendreau et. al., 1994), ant algorithm (Bullnheimer et. al., 1997; Gambardella et. al., 1999) and genetic algorithm (Berger and Barkaoui, 2003; Jih and Hsu, 1999; Thangiah et. al., 1991).

Our work considers the formulation for a CVRP, a variant of the VRP whereby the capacity of each vehicle is limited, although it should be emphasized that the CVRP can be subjected to further practical constraints in real-life supply chain management. An evolutionary algorithm was proposed as the problem-solving algorithm and we refer to the whole distribution planning system as the fuel truck dispatch system (FTDS). The scenario of the problem describes the routing of a fleet of dispatch trucks for distributing fuel to refuelling stations located over the whole of Singapore.

The rest of the paper is organized as follows. Section 2 describes the practical scenario of the fuel truck dispatch system (FTDS). Details on the architecture of the evolutionary algorithm are presented in Section 3. Simulation results based on dataset for refuel stations in Singapore are presented and discussed in Section 4. Finally, Section 5 concludes this paper with discussion on the outlook of the evolutionary algorithm in real-time supply chain management.

2. FTDS DESCRIPTION

We consider an application of VRP in a supply distribution network. Specifically, the problem deals with the distribution of fuel for the purpose of replenishing the supply at retail outlets. The system generates a plan for routing the various fuel trucks being managed. The trucks are sent to various fuel stations to replenish supply. The dispatch schedule is subject to a series of constraints described below.

All the trucks originate from the same terminal station. They are sent out to service a number of refuel stations according to the planned sequence assigned to them. The trucks return to the terminal at the end of sequence. Upon arrival at each station, the truck dispenses a fixed amount of fuel allocated to the station, or any lesser amount that the station may be able to take in at the moment. The main goal of planning is to minimize the total distance travelled by all the

trucks. From the description so far, it is clear that the problem to be solved in FTDS belongs to the class of capacitated VRP (CVRP).

In real life dispatch planning, there may be many constraints that the system has to deal with. Some of the likely constraints are described below.

- A fuel station may require more than one type of fuel. However, each truck is allowed to carry only one particular type of fuel in one dispatch.
- The planning should be flexible enough to handle certain priority cases. Priority may be accorded for situations, including extreme shortage of fuel, projected depletion of fuel, operator imposed priority and so on.
- Typically, fuel delivery is to be completed within a certain time limit. Certain timing elements such as truck speed, fuelling time, rest stop may also be considered.
- Trucks are allowed to make multiple trips from the terminal station.
- The number of trucks may vary.
- The system should be upward scalable in terms of the number of stations.

The complexity of the FTDS without handling constraints is itself challenging. It is therefore envisaged that real-life practical constraints imposed during planning may make the problem even more complex.

3. EVOLUTIONARY ALGORITHM

From an optimization point of view, the FTDS can be formulated as a CVRP. To solve the CVRP, we configured an evolutionary algorithm. The basic block architecture of the evolutionary algorithm is as shown by the flowchart in Figure 2.

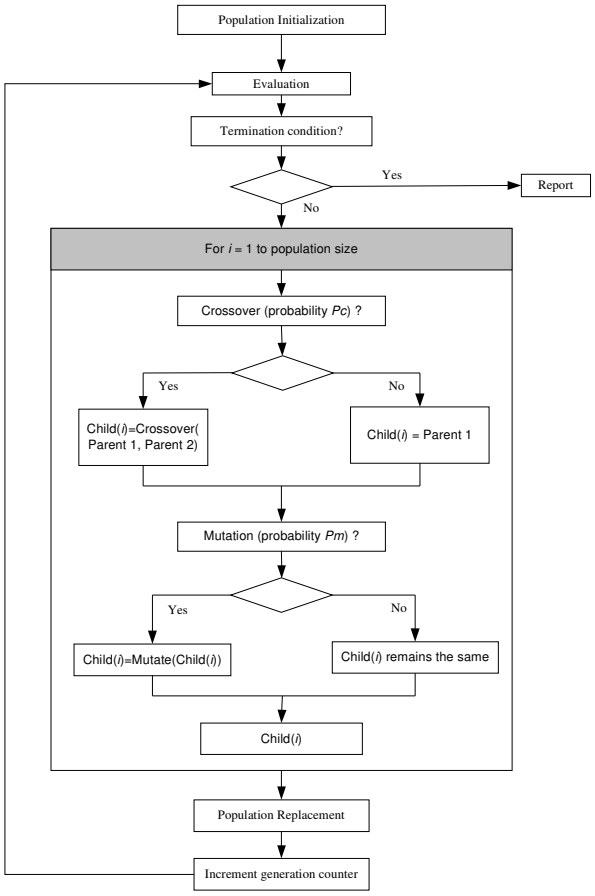

Figure 2. Flowchart of the evolutionary algorithm

The structure of Figure 2 is similar to the earlier reported hybrid genetic algorithms in Lim, Yuan and Omatu (Lim et. al., 2000; Lim et. al., 2002) for solving the quadratic assignment problems. First, an initial population of chromosomes representing possible solutions is randomly generated. After evaluation, a population of offspring is generated by crossover and mutation operations with pre-specified probabilities. Then, the whole population is replaced by the offspring except for a pool of elites which survive into the next generation. This process is repeated until the termination condition is met. The solution with the minimum distance is reported. In cases where a feasible solution is not possible, the system reports the best possible routing configuration to enable the operator to gauge the extent of the adjustment needed (such as increasing the capacity of a truck) in order to achieve a practically feasible plan. The remaining of this section describes the coding, formulation of the objective and fitness function, and the genetic operations.

3.1. Solution Representation

Given that there are S number of stations and T number of trucks. In our coding scheme, each chromosome is a possible routing plan for the fleet of vehicles. It is represented as an integer string of length S. Each allele represents the identity of a truck with an integer value from the set $\{1,2,\ldots,T\}$, while its position (from 1 to S) represents the station to be serviced by the truck. The stations (positions) within the string with the same truck (value of alleles) form the routes for this truck. For example, given that there are 6 stations (excluding the terminal station) and 2 trucks, from the chromosome as illustrated in Figure 3, Truck_1 services stations 1, 3 and 4 while the Truck_2 services the remaining stations. The coding scheme is elegant because of its simplicity. Besides allowing for easy representation of all possible plans of dispatching by a chromosome in the form of an integer string, it makes it easier to configure genetic operations to manipulate the strings during the evolutionary search process.

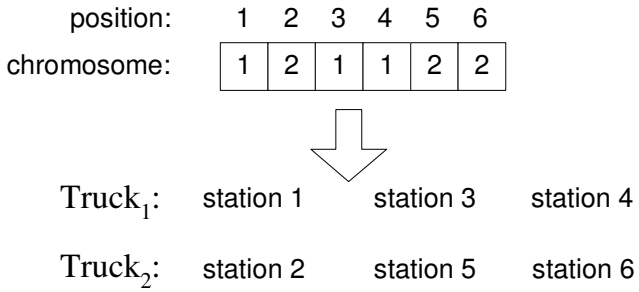

Figure 3. Solution representation

3.2. Objective Function

The objective of the search is to minimize the overall distance travelled by all the trucks. Let R denote the routing plan for a fleet of vehicles decoded from a genetic string. The objective function for the search can be written as Eq. (1) below:

$$dist(R) = \sum_{i=1}^{T} dist(Truck_i) \tag{1}$$

In the above equation, $dist(Truck_i)$ returns the distance travelled by the i-th truck in its tour, and $dist(R)$ represents the total distance travelled according to plan R. From the distance computed, the raw fitness value of a chromosome that results in a feasible routing plan can be calculated as follows:

$$raw\_fitness(R) = \frac{C}{dist(R)} \tag{2}$$

where C represents a numerical constant.

However, our algorithm does not exclude infeasible solutions during the search. If the routing plan is such that there is insufficient fuel for any trucks to fulfil the demand of all the stations it is to service, the resulting plan is deemed to be infeasible. To handle such situations, we incorporate penalty into the fitness function. The penalty applied to an infeasible solution is proportional to the extent of the shortfall of fuel for the corresponding truck. Eq. (3) represents the overall fitness function for the evolutionary algorithm incorporating the penalty component for infeasible solutions.

$$fitness(R) = raw\_fitness(R) - \lambda \sum_{i=1}^{T} \frac{shortfall(Truck_i)}{capacity(Truck_i)} \tag{3}$$

In Eq. (3), $shortfall(Truck_i)$ represents the extent of shortage for the i-th truck with respect to the overall demand of the stations that it has to service, and $capacity(Truck_i)$ represents the capacity of the i-th truck and λ is the weighting factor for the penalty to be applied.

We have earlier experimented with algorithm that configures only feasible solutions during the search. This is akin to complete penalty for infeasible solutions. Compared to applying complete penalty, i.e., discarding chromosomes that result in plans that are not feasible, our simulations have shown that using the fitness function as per Eq. (3) offers two advantages. First, it preserves those chromosomes with good objective values although infeasible, especially during the early stages of the search. This way, their good "attributes" are therefore inherited by their offspring, increasing the chance of producing good yet feasible solutions. The second advantage of using penalty incorporated fitness function is one of direct practical relevance. For situation where there is no feasible solution or the algorithm fails to produce any feasible solutions, solutions with the lowest distance although infeasible can still be generated. These solutions are still meaningful and useful, especially for further evaluation in light of the practical requirements of the customers.

3.3. Distance Calculation

In Eq. (1), the calculation of the shortest distance travelled by a truck is essentially the same as a travelling salesman problem (TSP). TSP is a very famous class of combinatorial optimization problems and is itself NP-hard. The general TSP involves finding the shortest circuit for a given set of nodes to be visited. In our approach, the accuracy and efficiency of the scheme for solving this routing problem has a great influence on the effectiveness of the global algorithm.

Our implementation adopts a layered strategy for calculating the lowest distance for each truck, with consideration of the trade-off between accuracy and efficiency. For the truck with no more than 6 stations to visit, we make use of an exhaustive enumerative method. It involves enumerating all the $(n-1)!/2$ possible routes to determine the sequence of visits that results in minimum distance, where n is the number of stations for a truck to visit (including the terminal station). Although such exhaustive method guarantees the optimal solution, the computational complexity is in the order of $O(n!)$. As n increases, the computing time tends to increase exponentially.

As an alternative for larger problems, we adopt a greedy search methodology. Firstly, based on the list of nodes that a truck is assigned to service, it starts the sequence by choosing from the list a station that is nearest to the terminal station. Then the next station in the sequence is determined by choosing the station that is nearest to the preceding station from the list of remaining stations. This process is repeated, until all the stations have been exhausted to form the complete sequence starting and ending at the terminal station. The complexity for this scheme is in the order of $O(n^2)$, and is much faster than the exhaustive enumerative method. However, the greedy search does not guarantee the optimal solution and the accuracy of it decreases as n increases. Consider d_{opt} and d_{greedy} to be the optimal distance and the distance obtained by greedy search respectively. In the worst case, the inaccuracy of the greedy search can be quantified according to Eq. (4) (Flood, 1956).

$$d_{greedy} / d_{opt} \leq 0.5(\lceil \log_2 n \rceil + 1) \quad (4)$$

From Eq. (4), it is clear that when n>16, the distance obtained by the greedy approach can be up to 3 times that of the minimum. To address such potential inaccuracy, larger instances of the problem may need to incorporate efficient heuristic methods, e.g. GA, to solve the TSP problem in a relatively short time and with acceptable accuracy. Successful examples of GAs for TSP optimization can be found in (Freisleben and Merz, 1996; Bentley, 1992). The overall concept of a layered decoding scheme for the CVRP is illustrated in Fig. 4.

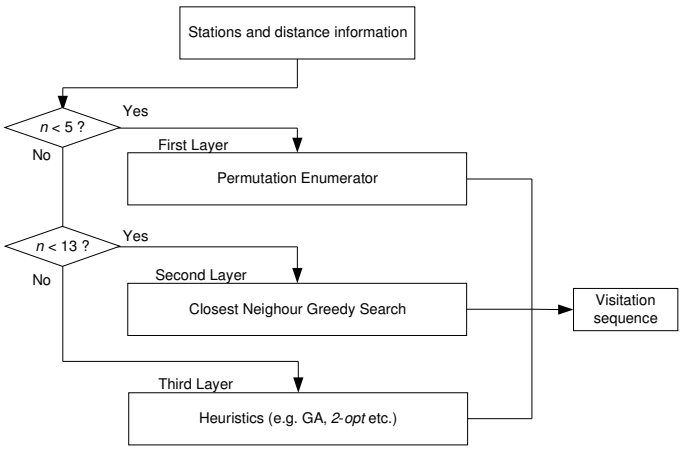

Figure 4. Layered decoding scheme

In Fig. 4, the first layer represents the most primitive decoding layer, applicable for problems where the number of nodes for each vehicle in the CVRP is small. The second layer is applicable for problems where the number of nodes is moderately large and inaccuracy in the estimation of the total distance travelled can be tolerated. For some problems, this may be insignificant since the dataset may also be subject to some form of inaccuracy. For large problems, computationally efficient approaches are required to compute the TSP optimization associated to each vehicle.

3.4. Genetic Operators

The importance of an effective coding scheme and the design of an appropriate fitness function for an evolutionary algorithm should be greatly emphasized. An elegant coding scheme can shield the complexity of the problem from the algorithm developer while a properly designed fitness function serves as an effective bridge between the real-world domain issues and the search algorithm. Together, both will facilitate the design of suitable genetic operations for exploring the domain search space. The following subsections outline the main genetic operations and mechanisms in our evolutionary algorithm approach.

3.4.1. Selection Scheme

The fitness measure is the basis for effective selection bias, a form of exploitation in evolutionary computation. With a mechanism for selection based on fitness, the search is biased in such a way that fitter chromosomes are accorded higher chance of survival, hence greater likelihood of them contributing towards the optimal solution. Based on considerations of the fitness function we have designed, biased roulette selection mechanism (Goldberg, 1989) can effectively create the necessary exploitative behavior. It assigns each chromosome proportionally higher chance of being selected for reproduction according to its fitness. Although we experimented with ranking and stochastic selection strategy, the biased roulette selection is by far the most effective in speeding up convergence towards good quality sub-optimal solutions.

3.4.2. Crossover

Crossover combines building blocks of good solutions from a population of diverse chromosomes with intention to create good quality offspring. In our implementation, we use a two-point crossover operation. The two-point crossover can be easily tailored to suit the problem's attributes and chromosome representation. An illustration of this scheme is as shown in Figure 5. In the figure, two crossing sites are randomly chosen. An offspring is produced by taking the segment between the two crossing sites from one parent, while the remaining part of the offspring comes from the second parent. Together with the selection scheme, the crossover can effectively create solution structures (assignment of trucks) with attributes acquired from fit parents. It is worth noting that one reason for the simplicity of the crossover operation can be attributed to the effective coding structure of the problem.

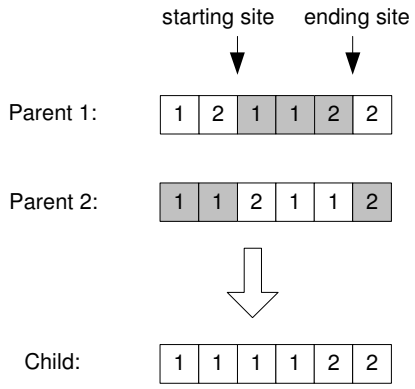

Figure 5. Two-point crossover

3.4.3. Mutation

Mutation causes random perturbation with intention to promote population diversity. It is also useful as a mechanism to shake the algorithm from a state of persistent local optima. With population diversity, the coverage of the search region is well spread out, enhancing the chance of finding good solutions. In our algorithm, mutation is achieved by random alteration of the allele of a gene. As shown in Figure 6, the third gene of chromosome *A* representing a routing plan of a 2-truck CVRP, is selected. Being a 2-truck system, the allelic code is therefore obtained from the set {1, 2}. Although the mutation involves only a single gene change, the resulting modification to the overall routing plan is significant. However, for problems where *n* (the number of vehicles) is large, the one-point mutation may not be sufficient to create the necessary population diversity. For this reason, we incorporated a parametric mutation operator which we refer to as α-mutation. The symbol α corresponds to the number of genes to be altered during mutation. Its value is pre-specified or can be approximated as *int(S / T)*, i.e., the whole number value of the ratio of the number of stations to the number of vehicles.

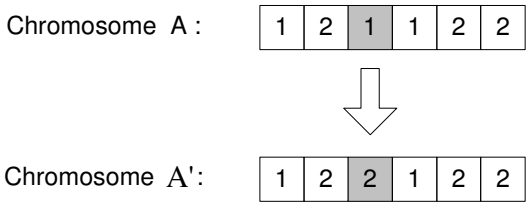

Figure 6. Mutation

3.4.4. Population Replacement and Elitism

For each generation, a new population is obtained by replacing the current population by offspring created through crossover. Besides the wholesale replacement, we adopt an elitist replacement scheme to preserve the best solutions obtained from one generation to another. Elitism is known to speed up convergence greatly. It simply duplicates a few best chromosomes from the current population into the new population. Several schemes can be considered in deciding which individuals in the new population to be replaced by the elite pool. One common scheme is to replace the worst performing chromosomes in the new population, an approach adopted in our implementation.

4. SIMULATION RESULTS

4.1. Testing on Benchmarks

We first test our evolutionary algorithm on several datasets of CVRP benchmarks (Christofides et. al., 1979) and compare it with some other competing VRP methods, in particular, the simulated annealing (Osman, 1993), the tabu search (Gendreau et. al., 1994) and the hybrid genetic algorithm (Berger and Barkaoui, 2003), referred to as OS, GHL and BB respectively. All simulations were done on a NEC PC with 2.4GHz CPU clock speed. For each instance, 10 independent trials were carried out to evaluate the average performance. The parameters of the evolutionary algorithm for our simulations are as follow:

Population size = 50;
Crossover rate = 1.0;
Mutation rate = 0.1;
Number of elites = 5;
Maximum number of generations = 100.

The results obtained compared against other approaches are summarized in Table 1. In Table 1, entries in the first column identify the instances of the benchmarks and their sizes (i.e. the number of stations). In the next 3 columns, the best results in term of travelling distance obtained by these three competing methods are presented. In the fifth column, we report the best and the average performance of our evolutionary algorithm and the last column shows the best-known values of the corresponding benchmarks.

According to the table, the results obtained by EA are competitive. For the smaller benchmarks, the best solutions of EA are comparable to the best solutions of the other approaches. Although in some instances, the best solutions obtained by the EA do not measure up to the results obtained by the other approaches, in particular the larger instances of the benchmark problems, the difference is not very significant. The following two points must be considered in order to make the analysis of the results more meaningful.

Table 1. Simulations on benchmark problems

Benchmark name (n)	OS	GHL	BB	Evolutionary Algorithm (EA)				Best-known
				Best		Average		
				Distance	Deviation	Distance	Deviation	
vrpnc1(50)	524.61	524.61	524.61	524.61	0.00%	525.27	0.13%	524.61
vrpnc2(75)	844.00	835.45	835.26	838.60	0.40%	845.46	1.22%	835.26
vrpnc3(100)	838.00	829.45	827.39	830.70	0.55%	845.63	2.36%	826.14
vrpnc4(150)	1044.35	1036.16	1036.16	1044.35	1.55%	1064.52	3.51%	1028.42
vrpnc5(199)	1334.55	1322.65	1324.06	1334.55	3.34%	1353.92	4.84%	1291.45

- The results of the other approaches (OS, GHL and BB) only include the best solutions obtained. It is therefore not possible to gauge the general performance of the algorithms. Our results for the EA include average and deviation based on multiple runs for each instance of the benchmarks and these values indicate that the performance of the EA is generally stable. This is a desirable characteristic for deployment in real-life application scenarios.
- The computational time costs of the other approaches were reported for different computing platforms. It is therefore difficult to analyze the performance of the algorithms objectively based on this criterion. Our EA converges to the desired solutions ranging from 20 seconds (n=50) to 16 minutes (n=199). We envisage that the computational cost can be further reduced by incorporating more powerful and diverse local search techniques.

In general, the smaller benchmarks are more in line with the application of the EA to solve the fuel dispatch problem. It is also noted that the EA has been crafted to address the realistic scenario of the supply chain management system. Unlike benchmarks where the optima are usually known, the EA in this algorithm is designed to handle scenarios whereby the optima are not known beforehand. It admits the possibility of the optimum being an infeasible solution as compared to benchmark problems where it is known a *priori* that the optimum is a feasible solution. This provides leverage for the algorithms designed for testing on benchmark problems to focus only on solutions that are feasible. Moreover, our EA is able to solve problems involving trucks of varying capacities, such as the VRP instances in the FTDS discussed below, while benchmark problems assume that all the trucks have the same capacity. This capability is more useful to suit the practical requirements of real-life applications.

4.2. Testing on FTDS dataset

The FTDS is simulated on dataset involving 30 refuelling stations island wide, including the terminal station. Table A1 in the Appendix shows the 30×30 distance matrix. Each entry in the matrix corresponds to distance between any two stations. Since the matrix is symmetric, only one half of the matrix is shown. Excluding the terminal station, which in this case is station 2, there are 29 refuelling stations to be serviced by a fleet of trucks. Typically at any time, not necessarily all the stations are involved in the planning as some may have ample supply and hence there is no need for replenishing of supply. Depending on the demand, it may not be necessary to deploy all the trucks at any one time. At the moment, we set the number of trucks as a *priori* knowledge although further extended work may involve generating plans that optimize on the number (or type) of trucks for deployment based on cost consideration.

We simulated the scenario where the terminal station has up to 5 trucks for deployment. At any time, a station may submit a request for a certain amount of fuel. Under normal circumstances, the capacity of all the 5 trucks is known beforehand. For our simulation purpose and to test the robustness of the algorithm, we allow for data on the capacity of the trucks to be entered as user input. The results of simulations for several input scenarios are presented in Tables 2 to 4.

In Table 2, we present the test results for what we termed as a loose configuration input or an over capacity situation. The total capacity of all trucks is significantly more than the total amount of fuel requested by the 29 stations. This did not present any difficulty and our algorithm was able to find a good routing plan with a total travelled distance of 214.0 km.

Table 2. A loose configuration

	Capacity (L)	Requested fuel (L)	No. of stations to visit	Distance traveled (km)
$Truck_1$	15000	13100	7	44.5
$Truck_2$	14000	9500	4	24.8
$Truck_3$	13000	12900	8	52.0
$Truck_4$	12000	11700	4	41.0
$Truck_5$	11000	10400	6	51.7
Total	65000	57600	29	214.0

Table 3 presents a sample of the simulation results for what we termed as a tight configuration input. This refers to the situation where the overall capacity of the trucks deployed is just about enough to meet the overall demand of the stations. To simulate this, the overall demand of fuel by all 29 stations is set to 1400 litres less than total capacity of all trucks. The FTDS was able to find a feasible plan with relative ease. As expected, the results shown in Table 3 show that each truck services a number of stations with total request for fuel not exceeding the capacity of the truck. In optimizing such tight configuration, it is not surprising that the minimal distance found is correspondingly more than that of the loose configuration scenario.

Table 3. A tight configuration

	Capacity (L)	Request fuel (L)	No. of stations to visit	Distance traveled (km)
$Truck_1$	13000	12900	7	65.0
$Truck_2$	12000	11900	5	69.8
$Truck_3$	12000	11000	6	46.7
$Truck_4$	11500	11300	7	73.1
$Truck_5$	10500	10500	4	34.0
Total	59000	57600	29	288.5

We tested a third scenario which corresponds to the under-capacity situation. To simulate this, we set the capacity of the trucks such that the overall capacity is lower than the total amount requested by the 29 stations. A straightforward way to handle this situation is for the algorithm to go through a preliminary check and then alert the operator that there is no feasible solution for the scenario. Although this may serve the purpose of the FTDS, it is not helpful to the operator as it does not give any indication whatsoever to help the operator in managing the situation. As such, we opted to design our algorithm such that it proceeds with the search and report the best possible infeasible solution to the operator. This is possible with the penalty based fitness function discussed earlier. Table 4 presents a sample of the results of simulation for an under-capacity configuration scenario. According to the results in Table 4, $Truck_1$, $Truck_2$ and $Truck_4$ violate the capacity constraint while $Truck_3$ and $Truck_5$ are able to utilize its capacity in a very tight manner. For situations pertaining to under-capacity which ultimately result in infeasible solutions, it is usually difficult to derive a standard measure as to what is perceived as a good infeasible solution. Typically, this can be determined by taking into account the practical considerations of the distribution network. Such considerations are then factored into the penalty component of the fitness function. In our case, the fitness function (see Eq. 3) is designed to favour solutions that attempt to spread out the excess proportionately. For example, in Table 4, $Truck_1$ with the highest capacity has the highest allocation of excess request.

Table 4. An under-capacity configuration

	Capacity (L)	Requested fuel (L)	No. of stations to visit	Traveled distance (km)
$Truck_1$	**13000**	**15000**	7	46.6
$Truck_2$	**12000**	**12700**	6	68.0
$Truck_3$	11000	11000	4	36.8
$Truck_4$	**10000**	**10100**	6	51.8
$Truck_5$	9000	8800	6	64.9
Total	55000	57600	29	268.1

A sample of the routing plan involving 5 trucks is depicted in Figure 7. The figure shows 30 stations spread across the island of Singapore. The routing of the 5 trucks is generated for a

tight configuration scenario, with a minimum distance of 288.5 km and a capacity to demand ratio of 1.024.

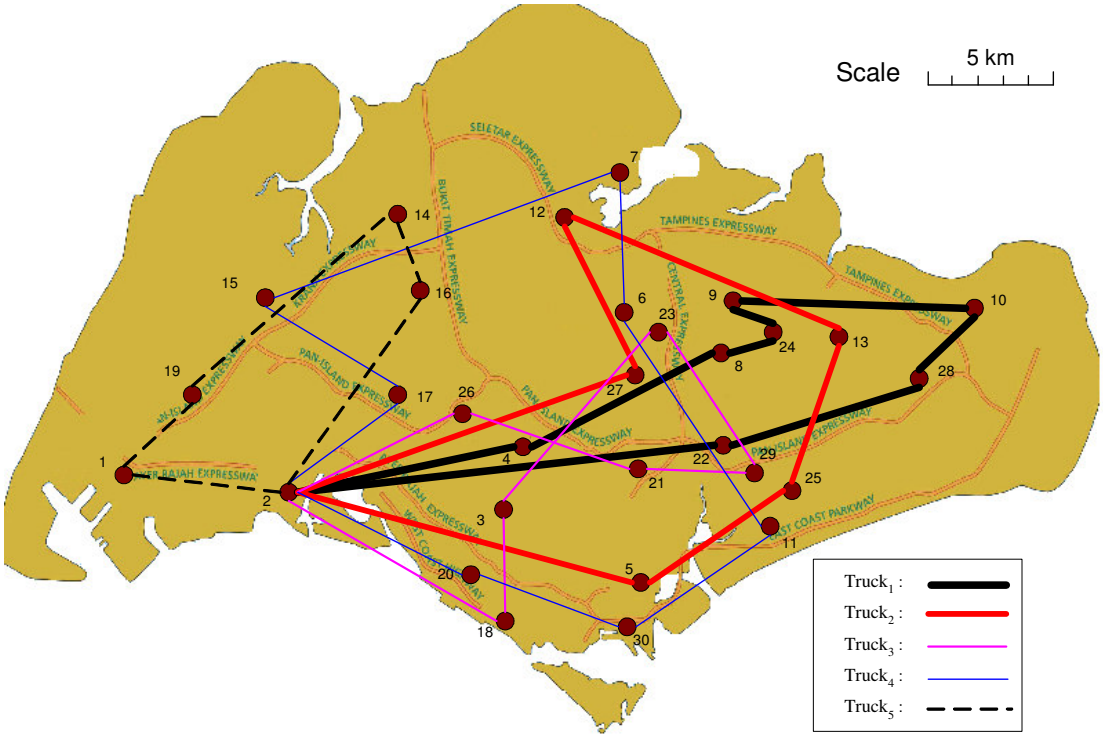

Figure 7. A sample routing plan for a tight configuration scenario

5. CONCLUSIONS

Real-life applications of stochastic algorithms need to consider trade-off in solution quality and computational efficiency. One such application is the problem of planning the dispatch of fuel in order to replenish the supply of refuelling stations, an important aspect of supply chain management. Our solution methodology is an evolutionary algorithm with a layered chromosome decoding strategy. The utilization of a 3-layered decoding strategy with criteria set out for each layer has been presented in this paper. In principle, the third layer decoding is a form of co-evolution similar to the approach we adopted in solving the curriculum time-table planning problem (Chan et. al., 2004). This will be further explored to extend the capability of the FTDS to solve problems of greater complexity. Beyond this, the co-evolutionary can be adapted to run in a distributed computing environment. So far, simulation results on various scenarios derived from a real-life dataset have demonstrated the robustness, efficiency and flexibility of our algorithm. The system generates good quality feasible solutions as well as the best possible infeasible solutions for cases of under-capacity.

REFERENCES

Badeau, P., Gendreau, M., Guertin, F., Potvin, J.Y. and Taillard, E. (1997) "A Parallel Tabu Search Heuristic for the Vehicle Routing Problem with Time Windows," *Transportation Research,* **5**, 109-122.

Bentley, J.J. (1992) "Fast Algorithms for Geometric Travelling Salesman Problems," *ORSA Journal on Computing*, **4**(4), 387-411.

Berger, J. and Barkaoui, M. (2003) "A Hybrid Genetic Algorithm for the Capacitated Vehicle Routing Problem, " *GECCO03,* Chicago, USA.

Bullnheimer, B.R., Hartl, F. and Strauss, C. (1997) "Applying the Ant System to the Vehicle Routing Problem," *2nd International Conference on Metaheuristics*, Sophia-Antipolis, France.

Chan, C. K., Gooi, H. B. and Lim, M.H. (2004) "Duration-Dependent Multi-Schedule Evolutionary Curriculum Timetabling," in Tan, K.C., Lim, M.H., Wang, L. and Yao, X. (eds.), *Recent Advances in Simulated Evolution and Learning*, World Scientific.

Christofides, N., Mingozzi, A. and Toth, P. (1979), "The Vehicle Routing Problem", in Christofides N., Mingozzi, A., Toth, P. and Sandi, C. (eds), *Combinatorial Optimization,* Wiley, Chichester 315–338.

Dantzig, G. B. and Ramser, R. H. (1959) "The Truck Dispatching Problem," *Management Science*, **6**, 80–91.

Fisher, M. L. and Jaikumar, R. (1981) "A Generalized Assignment Heuristic for Vehicle Routing," *Networks*, **11**, 109-124.

Fisher, M.L. (1994) "Optimal Solution of Vehicle Routing Problems Using Minimum K-trees," *Operations Research*, **42**, 626-642.

Flood, M.M. (1956) "The Traveling Salesman Problem," *Operations Research,* **4**, 61-75.

Freisleben, B. and Merz, P. (1996) "Genetic Local Search Algorithm for Solving Symmetric and Asymmetric Traveling Salesman Problems," *Proceedings of IEEE International Conference on Evolutionary Computation, IEEE-EC 96, IEEE Press*, 616-621.

Gambardella, L. M., Taillard, É. D. and Agazzi, G. (1999) "MACS-VRPTW: A Multiple Ant Colony System for Vehicle Routing Problems with Time Windows," in Corne, D., Dorigo, M. and Glover, F. (eds.), *New Ideas in Optimization,* McGraw-Hill.

Gendreau, M., Hertz, A. and Laporte, G. (1994), "A Tabu Search Heuristic for the Vehicle Routing: Problem", *Management Science,* **40***,* 1276–1290.

Goldberg, D.E. (1989) *Genetic algorithms in search, Optimization and Machine Learning*, Addison-Wesley, MA.

Jih, W. and Hsu, J.Y. (1999) "Dynamic Vehicle Routing Using Hybrid Genetic Algorithms," *In Proceedings of the 1999 IEEE International Conference on Robotics & Automation*. Detroit, Michigan.

Kelly, J. and Xu, J. P. (1996) "A Network Flow-Based Tabu Search Heuristic for the Vehicle Routing Problem," *Transportation Science*, **30**, 379-393.

Lim, M. H., Yu, Y. and Omatu, S, (2000) "Efficient Genetic Algorithms using Simple Genes Exchange Local Search Policy for the Quadratic Assignment Problem," *Computational Optimization and Applications (Kluwer Academic Publishers)*, **15**(3), 249-268.

Lim, M. H., Yu, Y. and Omatu, S. (2002) "Extensive Testing of A Hybrid Genetic Algorithm for Solving Quadratic Assignment Problems," *Computational Optimization and Applications (Kluwer Academic Publishers)*, **23**, 47-64.

Osman, I.H. (1993), "Metastrategy Simulated Annealing and Tabu Search Algorithms for the Vehicle Routing Problem", *Annal of Operations Research,* **41**, 421–451.

Taillard, É. D., 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,* **31**, 170-186.

Thangiah, S. R., Nygard, K. E. and Juell, P. L. (1991) "Gideon: A Genetic Algorithm System for Vehicle Routing with Time Windows," *In Proceedings of the Seventh Conference on Artificial Intelligence Applications*, 322-325, Miami, Florida.

Appendix

Table A1. Distance matrix of refuelling stations

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
1	x	5.2	16.3	19.0	24.3	28.7	27.8	30.8	33.6	42.2	31.5	27.3	38.0	14.2	9.1	16.2	14.5	19.0	4.7	15.9	19.9	28.4	28.1	31.5	33.3	18.4	25.1	38.6	31.7	23.2
2		x	6.7	9.7	14.9	18.0	27.5	20.2	22.5	31.1	22.1	22.5	25.9	11.5	11.7	12.2	5.6	7.3	7.6	6.5	10.9	17.3	15.4	20.4	18.9	6.5	13.3	27.5	21.1	13.8
3			x	3.2	7.1	11.1	17.1	13.3	15.6	24.2	13.3	15.6	20.0	10.3	15.5	10.6	6.8	4.5	14.2	1.8	3.4	10.4	10.5	13.5	15.3	4.8	7.6	20.6	13.7	7.6
4				x	8.2	9.1	15.1	11.3	13.6	22.2	12.4	13.6	18.0	9.0	13.8	10.1	6.4	6.8	14.5	5.3	1.4	8.4	8.5	11.5	13.9	3.0	5.6	18.6	11.7	8.8
5					x	11.6	18.3	10.2	11.6	18.8	7.2	16.1	14.6	17.2	23.3	16.7	15.3	7.1	21.5	8.6	5.9	6.4	11.3	9.5	8.2	11.0	8.6	15.2	7.5	2.0
6						x	6.8	6.6	7.0	18.5	11.7	4.6	17.3	11.8	18.7	15.5	15.0	14.4	23.2	12.9	7.6	7.7	2.5	6.5	12.6	11.3	4.4	17.9	11.04	13.1
7							x	11.5	12.2	19.2	16.6	3.6	19.6	10.2	17.1	14.0	17.5	20.7	23.6	19.2	13.9	12.6	7.1	11.4	17.5	17.6	9.1	20.4	15.9	20.2
8								x	1.9	12.6	5.7	12.7	10.2	18.4	24.5	17.9	17.4	15.5	25.7	18.1	9.4	4.1	5.0	0.8	6.6	13.7	2.9	12.1	5.1	13.2
9									x	11.8	7.2	12.3	10.5	19.4	26.3	19.4	18.9	17.0	27.2	19.6	10.9	5.6	4.7	1.3	7.6	15.2	4.4	12.1	6.6	14.7
10										x	12.1	19.9	3.5	27.0	33.9	29.3	28.8	25.7	37.0	26.5	19.8	15.4	15.8	11.0	10.9	24.9	17.2	2.5	11.8	20.2
11											x	18.2	7.1	5.1	26.1	19.5	18.9	14.4	28.8	16.0	8.2	3.6	9.6	5.8	0.6	14.9	7.8	8.8	1.8	8.8
12												x	18.5	8.7	15.6	12.5	18.5	19.6	24.0	17.1	12.1	13.5	7.0	11.0	17.0	15.4	9.2	20.0	15.1	21.8
13													x	24.7	31.5	24.2	23.5	21.4	31.7	20.5	14.2	8.8	14.2	8.5	5.7	19.6	12.4	0.9	7.3	17.0
14														x	4.9	1.1	5.0	14.3	10.9	10.7	12.4	16.2	14.2	18.0	19.7	6.5	13.3	27.3	20.4	20.4
15															x	6.6	8.9	18.8	8.0	15.8	18.9	22.7	23.1	25.8	27.6	12.1	19.5	32.9	26.0	23.0
16																x	5.3	14.6	12.9	10.9	11.9	15.7	15.0	18.1	19.2	6.7	12.7	25.2	18.4	18.2
17																	x	10.2	8.4	4.8	9.7	14.9	14.5	18.0	19.7	3.9	12.4	25.1	18.2	14.4
18																		x	16.3	3.5	8.0	12.1	13.9	15.9	15.0	10.1	11.8	22.9	13.9	6.1
19																			x	13.3	17.8	23.1	22.7	26.2	27.0	12.0	20.6	33.3	26.4	20.5
20																				x	7.9	14.7	12.3	14.9	15.1	4.6	10.2	22.0	15.1	7.7
21																					x	5.1	7.0	8.6	9.7	0.0	4.9	15.8	8.9	6.8
22																						x	5.6	3.5	4.3	11.0	3.8	10.4	3.5	7.8
23																							x	4.4	10.5	10.8	1.2	15.8	9.0	13.2
24																								x	5.5	14.0	4.3	11.2	4.7	12.8
25																									x	15.8	8.7	7.3	1.5	8.7
26																										x	7.4	21.1	14.3	11.1
27																											x	14.0	7.1	11.4
28																												x	8.5	17.8
29																													x	8.6
30																														x