Proceedings of the 2002 IEEE
International Symposium on Intelligent Control
Vancouver, Canada • October 27-30, 2002

Using Family Competition Genetic Algorithm in Pickup and Delivery Problem with Time Window Constraints

Wan-rong Jih Cheng-Yen Kao Jane Yung-jen Hsu

Department of Computer Science and Information Engineering
National Taiwan University
Taipei 106, Taiwan
jih@agents.csie.ntu.edu.tw cykao@csie.ntu.edu.tw yjhsu@csie.ntu.edu.tw

Abstract— **In this paper, we propose a novel research scheme to solve the single vehicle pickup and delivery problem (PDPTW) with time window constraints. The family competition genetic algorithm (FCGA) is a modern approach that has been successfully applied to solve the traveling salesman problem. We illustrate the family competition GA and give the experimental results that show the FCGA is a brilliant algorithm for solving the single vehicle PDPTW.**

Genetic algorithms (GA) have been successful applied to solve the combinatorial computation problems. The family competition will improve the achievements for obtaining optimal solutions and the probability to hit the feasible solutions. By comparing FCGA with traditional GA, this excellent approach does not need enormous resources. Applying FCGA to single vehicle PDPTW, it succeeded in finding feasible solutions for all problems and obtained efficient results in our experimentation.

I. Introduction

The pickup and delivery problem (PDP) is a NP-hard problem[7] and has been extensively studied in computer science, management and operational research. Consequently, the domain space of this problem grows exponentially. Under such time explosion, exact algorithms can only achieve an optimal solution with in a very small problem size. Therefore, the majority of the related researches apply heuristic algorithms to solve the pickup and delivery problems.

Many practical pickup and delivery problems had been applied to transportation of goods, robotics and the job-shop scheduling[1]. This problem is similar to Traveling Salesman Problems (TSP), Dial-a-Ride Problems (DARP) and the Vehicle Routing Problems (VRP). Except for its augmented constraints. For the detail discussion of pickup and delivery problems, readers might refer to the survey paper by Savelsbergh[10].

Genetic algorithms (GA) have been applied to solve the TSP[4],[8] and VRP[12],[2],[5],[6] successfully. Given this strength in routing related problems, we present a scheme that employs family competition genetic algorithms (FCGA) to solve the single vehicle pickup and delivery problem with time window constraints (PDPTW). The algorithms for single vehicle PDPTW are used as subroutines in multiple vehicles cases. The approaches for solving single vehicle PDPTW would be the principal algorithms on finding solutions of the multiple vehicles problems.

The organization of this paper is as follows. In the next section, we will introduce the single vehicle PDPTW. The chromosome representation and the fitness function of the FCGA will be given in Section . An outline of FCGA will show in section IV. The related operators of FCGA that will be used to solve the single vehicle PDPTW are described in section IV. Section V and section VI depict the experimental results and conclusions, respectively.

II. The Pickup and Delivery Problem

In the single vehicle pickup and delivery problem (PDP), a route must be constructed in order to satisfy transportation requests. Each transportation request specifies the weight of the load to be transported, a location where it is to be picked up and another location where it is to be delivered. Suppose that the beginning of a route is a depot where the vehicle departs from it. By starting from the depot, the vehicle travels through all the locations where the transportation requests specified. After the vehicle has visited all the transportation requests, the vehicle will park at one of the delivery location.

Pickup and delivery problems with time constraints (PDPTW) are strictly harder than the fundamental problems. In addition to the intrinsic precedence constraints and the vehicle capacity constraints, time constraints complicate the problem significantly. A time window is associated with every pickup location and the delivery location, which indicates the time interval that the corresponding location is available for service. The vehicle should visit every location and satisfy the given service time interval of the locations. If the vehicle arrives early, the vehicle has to wait at the arrived location until it is open.

A. Single vehicle PDPTW

Suppose that $N = \{1, \ldots, n\}$ is a set of transportation requests, where n is the number of transportation requests. To accomplish each transportation request $i \in N$, the vehicle should service the pickup location i^+ before the delivery location i^-. Accordingly, $V^+ = \{i^+ \mid i \in N\}$ and $V^- = \{i^- \mid i \in N\}$ are the set of pickup locations and delivery locations respectively. Let $G = (V, A)$ be a di-

rected graph where $V = \{0\} \cup V^+ \cup V^-$ is a set of locations, which includes an initial depot 0 and the specified location sets of the n transportation requests. Set $A = \{(r, s) \mid r \neq s, r, s \in V\}$ is a set of arcs. To measure the arrival time of each step, a given non-negative travel time d_{rs} is assigned to each arc $(r, s) \in A$. Each transportation request i is associated with a positive number q_i; it indicates the load of the transportation request. A vehicle should not be allowed to exceed the vehicle capacity Q.

In addition to the precedence constraints and the capacity constraints, PDPTW must also satisfy the time window constraints. For every transportation request i, are required to pick up the goods within the time interval $[a_{i^+}, b_{i^+}]$, and transport the goods to its delivery location i^- on the time interval $[a_{i^-}, b_{i^-}]$.

The time windows $[a_{i^+}, b_{i^+}]$ and $[a_{i^-}, b_{i^-}]$ are the available time interval of the pickup location i^+ and delivery location i^-, respectively. If the vehicle arrives in location $r \in V^+ \cup V^-$, and its arrival time t_r is earlier than the lower bound of the allowed arrival time a_r, it has to wait. That is, the departure time of a vehicle at location r will be equal to $max\{a_r, t_r\}$.

Every transportation request $i \in N$ is subject to $a_{i^+} \leq b_{i^+}$, $a_{i^-} \leq b_{i^-}$ and $b_{i^-} - a_{i^+} \geq d_{i^+i^-}$. The first two constraints declare the upper bound and lower bound of the time windows; the value of upper bound should higher than its lower bound. Constraint $b_{i^-} - a_{i^+} \geq d_{i^+i^-}$ corresponds to formula $a_{i^+} + d_{i^+i^-} \leq b_{i^-}$. It means the time windows for every transportation request $i \in N$ should be admissible. Given a transportation request $i \in N$, if a vehicle departs from a pickup location i^+ at time a_{i^+}, and its immediate destination is location i^-, which is the delivery location of the transportation request i. The vehicle will arrive location i^- at time $a_{i^+} + d_{i^+i^-}$, where $d_{i^+i^-}$ is the traveling cost between location i^+ and location i^-. Accordingly, value of $a_{i^+} + d_{i^+i^-}$ should less than the upper bound of time window b_{i^-}.

B. Goal

Initially, we assume that the vehicle is parked on an initial depot 0. When the transportation request $i \in N$ is assigned, the weight q_i of this transportation request i is known *a priori*. The value of travel time d_{rs} is pre-defined, where $r, s \in V$. Each transportation request i is associated with a pickup location i^+ and a delivery location i^-, besides, each location is associated with a time window.

Our goal is to find a vehicle route starting from an initial depot, finishing all transportation requests, and end at one of the delivery locations. According to the path, both the total traveling time and the total waiting time of the vehicle are minimized. Certainly, the path must be a feasible route that satisfies the capacity and the time window constraints of every transportation requests. Moreover, the feasible routes should satisfy the precedence constraints, that is, for every transportation request $i \in N$, the vehicle should visit its pickup location i^+ prior to the delivery location i^-.

III. Construction of Genetic Algorithm

Genetic algorithms are search algorithms based on the mechanics of natural selection and natural genetics. The "fittest" string structures are recombined with a structured yet randomized information exchange to form such a search algorithm. Genetic algorithm is a superior approach to solve the NP-hard problems[7].

To solve a problem by the genetic algorithm, we should define the representation of each chromosome first. According to the chromosome representation, an evaluation function will be defined. Conventionally, in genetic algorithm, the evaluation function is also named fitness function. The chromosome representation and fitness function play important roles in the design of genetic algorithm. Most parents selection methods choose individuals according to the their fitness value The recombination operators should be decided while the chromosome representation has been given.

A. Chromosome Representation

Typically, the representation for a route is a sequence of the location labels, to list the successional labels that a vehicle is visited. Suppose that the transportation requests $N = \{1, 2, 3\}$, a route $0 \rightarrow 3^+ \rightarrow 1^+ \rightarrow 1^- \rightarrow 2^+ \rightarrow 2^- \rightarrow 3^-$ the corresponding chromosome representation will be $(0\ 3^+\ 1^+\ 1^-\ 2^+\ 2^-\ 3^-)$. This representation is intuitively by following the traveling sequence in order, and it is easy to understand. The permutation representation is simple and easy to manipulate. If the transportation requests $N = \{1, \ldots, n\}$, by simply counting the pickup and delivery locations of every transportation request and a initial depot, the length of every chromosome will be $2n + 1$. Consequently, for solving the order-based problems, the permutation representation is the most common and popular representation.

B. Fitness function

Solutions to the single vehicle pickup and delivery problems with time window constraints are subject to the following constraints[3].

(1) Initially, a single vehicle parks on the initial depot 0, and it will visit all the specified locations exactly once.

(2) Every location $i^+ \in V^+$ should be visited before its delivery location $i^- \in V^-$. Such consideration is called precedence constraints.

(3) The capacity constraints represents the total load of a vehicle cannot exceed its capacity Q.

(4) Suppose that a vehicle arrives in the location $i \in V^+ \cup V^-$ at time t_i, the criterion $t_i \leq b_i$ must be satisfied. In general, we named this constraint the time window constraint.

(5) The vehicle routes will be the open paths, ending at any one of the delivery locations.

Given a chromosome, which represents a route S, the corresponding fitness function is defined in equation 1.

$$\Phi(S) = f_{travelcost}(S) + f_{penalty}(S) \quad (1)$$

The first part of function $\Phi(S)$ represents the total travel time for a vehicle to complete the route S. The value of $f_{travelcost}(S)$ will includes the waiting time while the vehicle arrives a location early.

The second part of Equation (1) is the penalty function of route S. A route might violate the constraints during the exploration of genetic algorithm. Therefore, a chromosome might represent an infeasible route. Function $f_{penalty}(S)$ defines the punishments for violating the constraints. If the vehicle arrives a location and it is overloading, penalization for violating the vehicle capacity will be given. Besides, penalties for delay of arriving a location will be adopted. Suppose that a vehicle arrives in location r at time t_r. If the arrival time $t_r < b_r$, there is a penalty for the vehicle arrives late. Symbol b_r stands for the upper bound of the time window $[a_r, b_r]$, which is associated with location r. The intrinsic precedence constraints should be strictly observed.

Given a chromosome, if the corresponding route S is a feasible route, the value of function $f_{penalty}(S)$ will equal to zero. Otherwise, for an infeasible route, there are penalizations, the value of penalty function certainly larger than zero. A feasible route denotes a route that might have waiting time but satisfy the precedence constraint, capacity constraint and time window constraint. If a route violates any constraints of the PDPTW, it is an invalid route, or namely the infeasible route.

Suppose that route S_i is an infeasible route and route S_f is feasible. According to the definition of $\Phi(S)$, the value of $\Phi(S_i)$ will much larger than that of the $\Phi(S_f)$. Consequently, we will try to find a feasible route S and the value of $\Phi(S)$ is the minimum. The goal of the single vehicle PDPTW is to minimize $\Phi(S)$ apparently.

C. Crossover

We will consider four crossover operators. The order crossover (OX) is an order-based crossover[8] and uniform order-based crossover (UOX) is the position-based crossover[11], both operators had been adapted to TSP during the past years. Merge cross #1 (MX1) and merge cross #2 (MX2) invented by Blanton[2] for solving VRP with time window constraints. The first two operators are traditional crossover operators and the last two operators use a global precedence vector to be the guidance of crossover.

Many studies had given exhaustive research on the first two traditional crossover operators. Therefore, in this paper, we will not describe the two traditional crossover operators. In regard to the MX1 and MX2, a primitive specification will be given on the following paragraph.

The key concept of merge crossover operators is based on a global precedence, an independent antecedence among the genes of the populations, that is, each gene in the chromosome has a precedence relation to every other gene. From the characteristics of constraints of the single vehicle PDPTW, a global precedence relation probably exists among genes. The global precedence vector is formed by such relationship and it could be the offspring-generating guidance.

In single-vehicle PDPTW, except the initial depot 0, every gene is either a delivery point or a pickup point and has an associated time window. We might utilize the time windows in order of precedence.

The operations of MX1 and MX2 are based on the global precedence vector. Operator MX1 produces a child that is close to the order of the global precedence. After perform operator MX2, will produce an offspring that the genes with lower priority will be moving to the end of the chromosome. For detail information related to merge crossover, readers might refer to the publications[2] of Blanton and Wainwright.

D. Mutation

In general, the mutation is worked with a single chromosome. A chromosome will be created by applying the mutation operator, and it will substitute the new chromosome for its original one. We will consider two mutation operators that can be applied to the single vehicle PDPTW problem.

The first mutation will select two genes randomly, and their positions are interchanged. This operator will create a new route, which has four different edges from its original route. The second mutation operator choose two cut sites randomly, and the order of the sub-route specified by the genes is inverted. Such mutation operator is the 2-opt move in TSP; the difference between the new and the original route is two edges.

IV. Family Competition GA

The family competition genetic algorithm (FCGA) is based on genetic algorithm (GA) with adding the concept of families. For every population, each individual owns its family. To maintain the constant size of a population, only the champion at a family survived.

The main concept of the family competition genetic algorithms (FCGA)[13] is similar to the rule of the natural evolution; the best solution is the only survivor from its family. Given a population t, a family contains u individuals and these individuals are the recombination of a family father F_i^t, where $i = 1, \ldots, m$ and m is the population size. Every individual I_i^t in population t has its turn to be the family father F_i^t.

In any generation t, each individual I_i^t in the current population P^t owns its family C_i^t. The individual F_i^t is the family father of family C_i^t and the size of the family is u. Accordingly, a generation t will produce $u \times m$ new individuals, where m is the population size.

Algorithm 1 elaborates the procedure of family competition genetic algorithm. The selection process for crossover operators is to select two parents; one is the family father F_i^t and the other is the alternative parent A_j^t. Alternative

Algorithm 1 The procedure of FCGA

1: $t = 0$;
2: Initialize population: $P^t \leftarrow \{I_1^t, I_2^t, \ldots, I_m^t\}$;
3: **Evaluation**: $P^t \leftarrow \{\Phi(I_1^t), \Phi(I_2^t), \ldots, \Phi(I_m^t)\}$;
4: **repeat**
5: $T \leftarrow \emptyset$;
6: **for** $i = 1$ **to** $\mid P^t \mid$ **do**
7: Family father: $F_i^t \leftarrow I_i^t$;
8: Family: $C_i^t \leftarrow \emptyset$;
9: **for** $j = 1$ **to** u **do**
10: **Selection**: alternative parent $A_j^t \in P^t$;
11: **Reproduction**: $c_j \leftarrow O_m(O_c(F_i^t, A_j^t))$;
12: **Evaluation**: $\Phi(c_j)$;
13: $C_i^t \leftarrow C_i^t \cup \{c_j\}$;
14: **end for**
15: $T \leftarrow T \cup best(C_i^t)$;
16: **end for**
17: New population P^{t+1} : $P^{t+1} \leftarrow P^t \cup T$;
18: $t = t + 1$;
19: **until** the termination condition is reached
20: Output the solutions;

parent A_j^t is randomly selected from the population P^t and is distinct from the family father F_i^t.

The procedure of reproduction will perform the recombination operations. The reproduction process might execute the crossover O_c and then perform the mutation O_m or simply execute the crossover. The family father F_i^t generates a new offspring c_j by using the recombination operators with a specific probability. The crossover rate is a fixed real number between 0 and 1, that is, analogous to the traditional genetic algorithm, using a fixed rate to perform crossover. In our FCGA, the occasion to execute the mutation depends on the similarity between the parents and its offspring. If the offspring is similar to one of its parents, perform the mutation. Accordingly, our FCGA will perform mutation while the offspring and one of its parents represents the same route. Such adaptive strategy of executing mutation supports the route diversity and prevents the search space be bound in local optimal solutions. It also precludes the algorithm from premature convergence.

In Algorithm 1, the evaluation function shown in the line 3 and 12 has been depicted in the previous section. The definition of function $\Phi(S)$ is consistent in this paper. The fitness function of the family competition genetic algorithms is function $\Phi(c_j)$, where c_j is a chromosome, which represents a route for the single vehicle pickup and delivery problems.

After the end of the inner for loop, the family C_i^t contains u individuals. Generally, most genetic algorithms support the fixed population size in the same run. to preserve the legitimately constant population, the individual who owns the best fitness value will be chosen from the family C_i^t. Given a family C, the function $best(C)$ will return an individual c_j, which has the minimum fitness value $\Phi(c_j)$. Accordingly, the new population size of T will be identical to the size of population P^t. Population P^t holds m individuals and the population T will hold the same individuals while the program executes the command after the end of the outer for loop. The individuals of the new population P^{t+1} are selected from the population P^t and population T.

Algorithm 1 will repeat on executing the new population generation procedure until the termination condition is reached. Finally, output result will equal to the best solution in the current population.

V. Experimental design and results

An algorithm for generating test sets had been developed, for obtaining the authentic datasets. The experimental results are use the datasets which generated by Algorithm 2.

A. Experimental design

Up to present, there is no suitable test sets for PDPTW. The datasets for PDPTW should concern with the precedence constraint, capacity constraint and time window constraint. The test data for TSP definitely not suit to the PDPTW because the datasets are not associated with any of the above constraints. As for the VRPTW instances, there are no precedence constraints. Therefore, we invent an algorithm to create PDPTW datasets for the experiments, shows in Algorithm 2.

Algorithm 2 Create test data for single vehicle PDPTW

Require: Give the number of task n, and $width$ is the width of time window
1: $N = \{1, \cdots, n\}$
2: **for** $i \in N$ **do**
3: Generate locations i^+ and i^- randomly;
4: Generate the demand q_i randomly;
5: **end for**
6: Evaluate the traveling time d_{rs}, $r, s \in V$;
7: Randomly generate a route which satisifies the precedence contraint;
8: Let t_i be the arrival time of location $i \in V$;
9: Suppose that the average traveling time between the locations is $AverageTime$;
10: **for** $i \in V^+ \cup V^-$ **do**
11: $a_i = t_i - (random(width) \times AverageTime$;
12: $b_i = t_i + (random(width) \times AverageTime$;
13: **end for**
14: $[a_i, b_i]$ is the time window for location $i \in V$;

B. Experimental results

A dynamic programming approach proposed that can achieves the optimal route and this dynamic programming had been elaborated on the publications of Psaraftis[9]. Use the dynamic programming to solve the single vehicle PDPTW, the execution time will explode, as the problem size increase. Furthermore, the dynamic programming method needs gigantic computer resources to execute and store the intermediate results. Therefore, most researcher

could not attain to optimal solutions while the task size is large, neither we are.

In our experiments, every test case had been performed crossover with 3 different crossover rate, said 45%, 60% and 75%. Moreover, given a specified crossover operator and the crossover rate, the experimentation would associate with a mutation operator. We apply the 2-point mutation and the 2-opt mutation operators to our experimentations. After determined the associated parameters of genetic algorithm, each case will be executed 30 runs.

Table I demonstrates the percentage above the optimal value by using the order-based crossover (OX) and merge crossover (MX1). Columns 2 to 4 represents the relative error of the best solutions to the optimum; if an item with value 0 means it is equal to the optimal solution. Symbol 'x' represents that no feasible solution had been found. The last column given the optimal solutions that obtained by using the Psaraftis[9] approach. Optimal values in parentheses stand for the best solution in our experimental results.

TABLE I
THE BEST SOLUTION RESULTS (PERCENTAGE OF RELATIVE ERROR) – OX & MX1

task size	relative error(%)				optimal value
	OX		MX1		
	GA	FCGA	GA	FCGA	
10	0.344	0.000	0.000	0.000	872
20	x	7.685	0.000	0.000	2030
30	x	x	0.000	0.189	3713
40	x	x	0.068	0.068	4386
50	x	x	0.017	0.017	(5752)
60	x	x	0.000	0.000	(5658)
70	x	x	0.429	0.249	(7221)
80	x	x	0.688	0.764	(7849)
90	x	x	0.046	0.313	(8618)
100	x	x	0.151	0.000	(10600)

Obviously, the order-based crossover (OX) is not suit for single vehicle PDPTW. Even the small problems OX still cannot get the feasible solutions. The merge crossover #1 (MX1) could not always obtain the optimal solution. Though FCGA improve the quality of the most solutions, for problem size 30 and 80, the fitness value is increased.

Figure 1 shown the best and average fitness value while the problem size is 80. The solid line represents the average fitness values that achieved by GA; it has the highest solution cost in every generation. Symbol FCGA(avg) indicates the average fitness values of FCGA. Figure 1 shows the average fitness values of FCGA are lower than that of GA, and both the two algorithms obtain the similar results of the best fitness values.

Table II shows the results of the UOX and MX2. Table I and Table II are refer to the same optimal values. Values in parentheses are the number of best value found under the trials. By applying the position-based crossover (UOX), the solution qualities of FCGA have significant improved. The FCGA might not promote the solution qualities while the transportation requests are less than 50. For larger

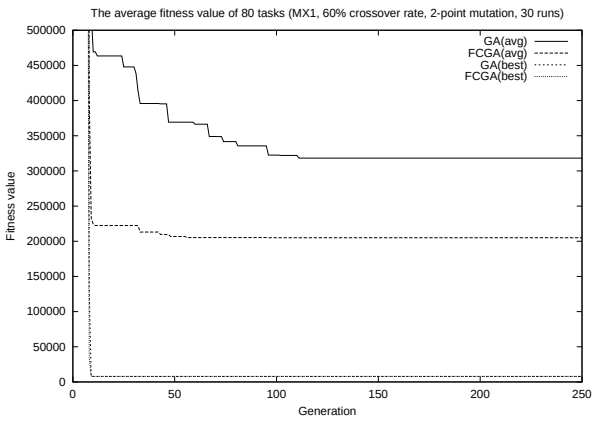

Fig. 1. Fitness values of MX1, 80 transportation requests

problem, FCGA achieve better results than GA.

TABLE II
THE BEST SOLUTION RESULTS (PERCENTAGE OF RELATIVE ERROR AND HIT RATIO) – MX1 & MX2

task size	relative error%(optimum/trials)			
	UOX		MX2	
	GA	FCGA	GA	FCGA
10	0.000(18/30)	0.000(27/30)	0.000(30/30)	0.000(30/30)
20	0.000(22/30)	0.000(30/30)	0.000(21/30)	0.000(11/30)
30	0.000(5/30)	0.000(20/30)	0.000(24/30)	0.000(7/30)
40	0.000(3/30)	0.000(23/30)	0.000(28/30)	0.000(12/30)
50	0.000(1/30)	0.000(17/30)	0.000(3/30)	0.017(24/30)
60	0.000(3/30)	0.000(5/30)	0.000(29/30)	0.000(30/30)
70	0.000(2/30)	0.000(19/30)	0.000(2/30)	0.000(2/30)
80	0.000(1/30)	0.000(4/30)	0.013(1/30)	0.000(1/30)
90	x	0.000(5/30)	0.000(1/30)	0.000(1/30)
100	0.000(1/30)	0.000(5/30)	0.000(10/30)	0.000(26/30)

Using UOX by the FCGA, the numbers of finding the best solution are increased substantially. For size is smaller than 50 request pairs, MX2 could not improve the best solution hit ratios. Since the best fitness value of size 50 could not improved by MX2, Figure 2 will show the best and average fitness values of MX2. Solid line is the average fitness values of GA, which converge to near the best solutions. However, the average fitness values of FCGA went prematurely. MX2 moves the least preferred gene to near the end of the route, this might cause the search exploration converged early while the problem size has not adequate spaces for the algorithms to search through.

Figure 3 depicts the average CPU time of FCGA and GA. The unit of the CPU time is second. Though the family competition will take more steps during the reproduction procedure of genetic algorithm, the execution time does not increase substantially. Study both the Table II and Figure 3, UOX obtained the best solutions by FCGA, whereas it required much time to achieve the solutions. The average execution time of MX1 is less than that of MX2, however, the solution qualities of MX2 are better than that of MX1.

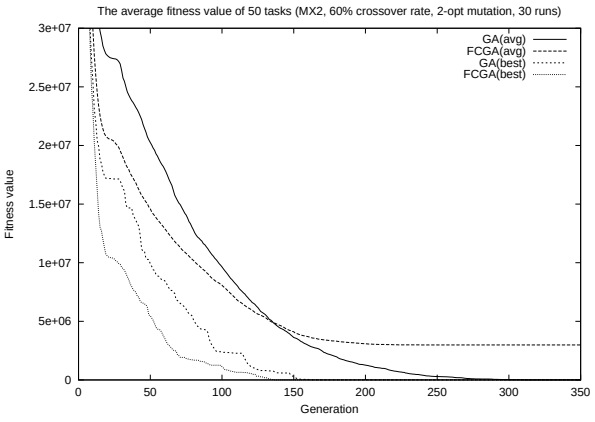

Fig. 2. Fitness value of MX2, 50 transportation requests

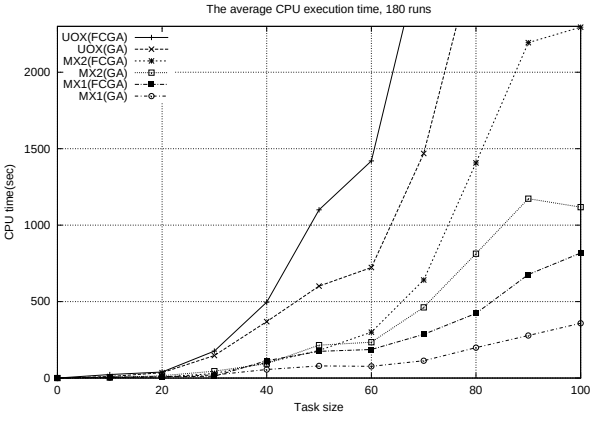

Fig. 3. A comparison of CPU time (UOX, MX1 and MX2)

In real-world applications, users should not be waiting too long for service. A feasible and near-optimal solution will be allowed to serve customers, whereas efficient and reliable response should be satisfied. Table III describes the percentage of the feasible solution hit ratio. FCGA improved the probability to obtain feasible solutions in all the three crossover operators. Under execution time consideration, MX1 and MX2 might suit for the real-time applications, by referring to Figure 3.

VI. Conclusion

The FCGA is a modern approach for solving the single vehicle pickup and delivery problems with time windows constraints. Every recombination operator has its particular characters but not all the reombination operators are suitable for solving the single vehicle PDPTW. We should exploit the experiment results to expand expertise. Understanding which operators are admissible operators for solving our problems will improve the probability of obtaining optimal solution.

The majority of genetic algorithms can obtain the optimal or near-optimal solutions. By comparing the solution cost of FCGA with that of traditional GA, in most cases, FCGA always improve the solution qualities of GA.

TABLE III
Percentage of feasible solution found, 180 runs

task size	feasible/trials(%)					
	UOX		MX1		MX2	
	GA	FCGA	GA	FCGA	GA	FCGA
10	100.00	100.00	100.00	100.00	100.00	100.00
20	98.89	100.00	100.00	100.00	100.00	100.00
30	70.00	93.33	100.00	97.78	98.89	98.33
40	45.00	90.00	100.00	100.00	99.44	96.11
50	8.33	46.67	100.00	100.00	99.44	84.44
60	5.00	15.56	98.33	100.00	96.67	87.22
70	7.22	47.22	100.00	100.00	97.22	95.00
80	11.67	11.67	21.67	55.00	61.11	80.56
90	x	8.89	61.11	96.11	71.67	87.22
100	1.11	13.33	7.78	26.67	25.00	63.89

We will continue to design the affiliated programs and recombination operators, so that we can accomplish the scheme and improve it. Furthermore, we hope that applications of this novel technique to the other practical science will achieve the similar outstanding results.

References

[1] A. A. Assad. Modeling and implentation issues in vehicle routing. In B. L. Golden and A. A. Assad, editors, *Vehicle Routing: Methods and Studies*, pages 7–45. Elsevier Science Publishers, North-Holland, Amsterdan, 1988.

[2] J. L. Blanton Jr. and R. L. Wainwright. Multiple vehicle routing with time and capacity constraints using genetic algorithms. In *Proceedings of the Fifth International Conference on Genetic Algorithms and Their Applications*, pages 452–459, 1993.

[3] M. Desrochers, J. K. Lenstra, and M. W. P. Soumis. Vehicle routing with time windows: Optimization and approximation. In B. L. Golden and A. A. Assad, editors, *Vehicle Routing:Methods and Studies*, pages 65–84. Elsevier Science Publishers, North-Holland, Amsterdan, 1988.

[4] A. Homaifar, S. Guan, and G. E. Liepins. A new approach on the traveling salesman problem by genetic algorithms. In *Proceedings of the Fifth International Conference on Genetic Algorithms and Their Applications*, pages 460–466, 1993.

[5] W. R. Jih, Y. P. Chen, and Y. J. Hsu. A comparative study of genetic algorithms for vehicle routing with time constraints. In *Proceedings of the 1996 International Computer Symposium*, pages 17–24, Kaohsiung, Taiwan, December 1996.

[6] W. R. Jih and Y. J. Hsu. Dynamic vehicle routing using hybrid genetic algorithms. In *Proceedings of the 1999 IEEE International Conference on Robotics & Automation*, pages 453–458, Detroit, Michigan, 1999.

[7] J. K. Lenstra and A. H. G. Rinnooy Kan. Complexity of vehicle routing and scheduling problems. *Networks*, 11:221–227, 1981.

[8] I. M. Oliver, D. J. Smith, and J. R. C. Holland. A study of permutation crossover operators on the traveling salesman problem. In *Proceedings of the Second International Conference on Genetic Algorithms and Their Applications*, pages 224–230, 1987.

[9] H. N. Psaraftis. Scheduling large-scale advance-request dial-a-ride systems. *American Journal of Mathematical and Management Sciences*, 6(3 & 4):327–341, 1986.

[10] M. W. P. Savelsbergh and M. Sol. The general pickup and delivery problem. *Transportation Science*, 29(1):17–29, 1995.

[11] G. Syswerda. Schedule optimization using genetic algorithms. In L. Davis, editor, *Handbook of Genetic Algorithms*, pages 332–349. Van Nostrand Reinhold, New York, 1991.

[12] S. R. Thangiah, R. Vinayagamoorthy, and A. Gubbi. Vehicle routing with time deadlines using genetic and local algorithms. In *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 506–513, 1993.

[13] J. M. Yang and C. Y. Kao. Integrating adaptive mutations and family competition into genetic algorithms as function optimizer. *Soft Computing*, 4(2):89–102, 2000.