Bulletin of the College of Engineering, N.T.U., No. 90, February 2004, pp. 89–98
國立臺灣大學「台大工程」學刊　第九十期　民國九十三年二月　第 89–98 頁

A FAMILY COMPETITION GENETIC ALGORITHM FOR THE PICKUP AND DELIVERY PROBLEMS WITH TIME WINDOW

以族群競爭式基因演算法解決具有次序及時間限制的載運路徑規劃問題

Wan-Rong Jih[*]　*Jane Yung-Jen Hsu*[†]
紀婉容[*]　許永真[†]

[*]Ph.D. candidate　[†]Professor
Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan 10617, R.O.C.
[*]博士候選人　[†]教授
國立台灣大學資訊工程研究所

摘　要

族群競爭式基因演算法 (FCGA) 是架構在傳統基因演算法上的新方法。根據已往的研究顯示，傳統的基因演算法能夠在合理的時間範圍內找到可行解，甚至是近似最佳解。FCGA 主要是在傳統的基因演算法中，加入族群競爭的觀念，使問題在求解過程中能夠更具有多樣性及變化，進而提高找到可行解的機率，最後並得到更好的結果。在本論文中，我們將利用 FCGA 來解決單一車輛在具有載運次序及時間限制的問題 (1-PDPTW)。 同時，藉由大量的實驗數據，在小於 50 組的工作排程上，驗證 FCGA 能有效的找到最佳解。而當動態規劃方法 (dynamic programming) 無法安排 50 組以上的工作排程時，FCGA 仍能有效的提供可行解給使用者。

關鍵詞： 取貨及送貨問題、載運路徑安排問題、時間限制、基因演算法。

Abstract

This paper presents a new approach based on genetic algorithms to solving the single-vehicle pickup and delivery problem with time window constraints (1-PDPTW). In particular, we illustrate how the Family Competition Genetic Algorithm (FCGA) is applied to the 1-PDPTW, and compare its results with previous solutions to the problem. Genetic algorithms have been shown to find feasible or near-optimal solutions when traditional methods would fail within a reasonable amount of time. By incorporating the concept of families to maintain diversity, FCGA further improves solution quality and increases the probability of finding the optimal solutions without much extra resource demands. Extensive experiments on FCGA with various crossover and mutation operators show that the new approach succeeds in finding the optimal solutions for 1-PDPTW under 50 tasks, and can obtain near-optimal or feasible solutions in most problems up to 100 tasks.

Keywords: pickup and delivery problem, vehicle routing problem, time windows, genetic algorithm.

1. INTRODUCTION

Many real-world transportation problems, such as vehicle dispatching [1,2] and winter gritting [3,4], can be formulated as the pickup and delivery problems (PDP) [5]. When the problem involves finding a set of minimum-cost routes for a fleet of vehicles to satisfy transportation requests with time constraints, it is called the pickup and delivery problem with time windows (PDPTW). Vehicles are assumed to depart from a

common depot. For each *transportation request*, or a *task*, loading and unloading shall be done within a particular time interval. It is also associated with a load, that is, the specific weight to be delivered. Each vehicle can carry a limited weight. A special class of PDPTW, the *1-PDPTW*, is to find the optimal route for a *single* vehicle to serve all the transportation requests.

The PDPTW belongs to the class of NP-hard problems [6], that is, the computation required to find a solution grows exponentially with its problem size. Psaraftis [7~9] proposed a solution using dynamic programming with a time complexity of $O(n^2 3^n)$, where n is the number of tasks. The proposed method solved PDPTW problems with up to 10 transportation requests, *i.e*., the solution route involves at most 21 location points. Desrosiers *et al*. [10] improved the original dynamic programming algorithm with elimination criteria, and solved the problems up to 40 tasks effectively. While faster CPUs have enable optimal solutions to be obtained for slightly larger problems, the algorithms still fail to find optimal solutions except for problems of relatively small size. Therefore, heuristic methods are designed to solve large problems of PDPTW on demand. Sexton *et al*. [11] formulated the problem as a linear program and decompose it into independent subproblems. Dumas *et al*. [12] employed the *column generation scheme* to solve 1-PDPTW for up to 55 paired requests. Comprehensive surveys have been compiled by Dumas [12], Savelsbergh [13] and Sigurd [14].

There are several well-known problems related to PDPTW. In the dial-a-ride problem (DARP), customers making the transportation requests are the load to be carried, and each customer is assumed to be of unit weight. DARP can be solved by dynamic programming as proposed by Psaraftis [7~9] and Desrosiers [10]. Furthermore, approximation algorithms for solving DARP were given in Madsen [15] and Charikar *et al*. [16] Surveys by Christofides [17] and Laporte [18] include both exact and heuristic cases to solving the traveling salesman problem (TSP). The traveling salesman problem with precedence constraints (TSPPC) imposes an ordering on the locations to be visited. Heuristic approaches have been developed by Anily [19], Gendreau [20] and Moon [21]. The vehicle routing problem (VRP) is a PDP, except that every task has been assigned to a common delivery location. Overviews of exact and approximate algorithms for VRP are provided by Psaraftis [22] and Laporte [23]. Readers may refer to Golden [24] Solomon [25] and Desrochers [26] for surveys on VRPTW, and a recent survey by Desaulniers *et al*. [27]. Laporte [28] collected a bibliography of routing related problems, including references on VRP and TSP.

Genetic algorithms (GAs) have been successfully applied to solve many combinatorial problems, including several types of VRP and TSP. Two edge-based recombination operators, alternate edges crossover and edge recombination crossover, were proposed by Grefenstette [29] and Whitley [30]. Starkweather [31] introduced the enhanced edge recombination crossover on TSP. He also compared many genetic operators including order crossover, order crossover #2 [32], partially-mapped crossover [33] and cycle crossover [34]. Homaifar [35] presented matrix representation and matrix crossover for solving the TSP. Freisleben and Merz [36,37] invented distance preserving crossover for TSP such that the offspring of this operator can escape the local optima in the search space. Surveys of GA on TSP were presented by Potvin [38] and Larranaga [39].

Blanton and Wainwright [40] proposed two crossover operators, merge crossover #1 and merge crossover #2, that utilize *global knowledge* to explore the solution space. Jih [41,42] performed a comparative study of GA with various recombination operators in solving PDPTW, and showed that the merge crossover operators are superior to the traditional ones. On the other hand, a new variation of GA called Family Competition Genetic Algorithm (FCGA) has been successfully applied to TSP and routing-related problems [43]. This research explores FCGA-based solutions to 1-PDPTW.

Section 2 presents the formal definition of 1-PDPTW, followed by the GA construction in terms of the chromosome representation, fitness function, and genetic operators. The FCGA algorithm is outlined in Section 4. Experimental results are summarized in Section 5, followed by the conclusions in Section 6.

2. PICKUP AND DELIVERY PROBLEM WITH TIME WINDOWS

In the single-vehicle pickup and delivery problem (1-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 from which the vehicle departs. Starting from the depot, the vehicle travels through all the locations where the transportation requests are specified. After the vehicle has fulfilled all the transportation requests, the vehicle will park at one of the delivery locations.

The single-vehicle pickup and delivery problems

with time constraints (1-PDPTW) are strictly harder than the basic 1-PDP problems. In addition to the intrinsic precedence and capacity constraints, the temporal constraints complicate the problem significantly. Each pickup and delivery location is associated with a time window, which specifies the service time interval allowed.

2.1 1-PDPTW

Let $N = \{1, \cdots, n\}$ represent the set of n transportation requests. For each task $i \in N$, the time windows $[a_{i+}, b_{i+}]$ and $[a_{i-}, b_{i-}]$ denote the available time interval of pickup location i^+ and delivery location i^-, respectively. A positive number q_i indicates the load of task i. While serving the transportation requests, the vehicle shall not exceed the capacity limit Q.

Given a directed graph $G = (V, A)$, let $V = \{0\} \cup V^+ \cup V^-$ be a set of nodes, where 0 is an initial depot. $V^+ = \{i^+ \mid i \in N\}$ is the set of pickup locations, and $V^- = \{i^- \mid i \in N\}$ is the set of delivery locations. The arc set is $A = \{(r, s) \mid r \neq s, r, s \in V\}$.

There are three types of constraints for 1-PDPTW: precedence constraints, capacity constraints and time window constraints. Assume that a vehicle is parked at the initial depot 0, and it shall visit all the specified locations exactly once. To accomplish a task $i \in N$, the vehicle shall serve the pickup location i^+ before the delivery location i^-; this is the *precedence constraint*. The *capacity constraint* states that the total load of a vehicle cannot exceed its capacity Q.

If the vehicle arrives at location $r \in V^+ \cup V^-$, its arrival time t_r should meet the criterion $a_r \le t_r \le b_r$, where $[a_r, b_r]$ is the time window of location r. However, if the arrival time t_r is earlier than the lower bound of the available time interval a_r, it has to wait until the time reaches a_r. That is, the departure time of a vehicle at location r will be equal to $\max\{a_r, t_r\}$. These criteria form the *time window constraints*.

2.2 Goal

Our goal is to find a vehicle route that starts from an initial depot, fulfills all the transportation requests, and ends at one of the delivery locations. The path should be a feasible route that satisfies the associated constraints, and minimizes the total traveling time and the total waiting time of the vehicle.

3. CONSTRUCTION OF GENETIC ALGORITHM

Genetic algorithms (GAs) are search algorithms based on the mechanics of natural selection and natural genetics. As in evolution, genetic algorithms utilize genetic recombination and replication on strings as the optimization procedures. The approach has been shown to be very effective in searching for solutions to NP-hard problems [6].

Deployment of any genetic algorithm requires defining the problem encoded in some *chromosome representation*, as well as the corresponding *evaluation function*, also called the *fitness function*. Parent selection methods often choose individuals from the current population based on their fitness functions. Given a chromosome representation, recombination operators are designed to create new individuals from the selected parents.

3.1 Chromosome Representation

A solution to 1-PDPTW will be represented as an ordered list of locations. Given a set of transportation requests $N = \{1, \cdots, n\}$, let i^+ and i^- denote the pickup and delivery location of task i, respectively. For instance, $(0\ 3^+\ 1^+\ 1^-\ 2^+\ 2\ 3^-)$ is the chromosome representation of route $0 \rightarrow 3^+ \rightarrow 1^+ \rightarrow 1^- \rightarrow 2^+ \rightarrow 2^- \rightarrow 3^-$. By simply counting the pickup and delivery locations of every transportation request and an initial depot, the length of every chromosome will be $2n + 1$. The simple permutation representation intuitively follows the travelling sequence, and it is the most common and popular representation for solving the order-based problems.

3.2 Fitness Function

Given a chromosome that represents a route S, the corresponding fitness function is defined in Eq. (1).

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

The value of $f_{\text{travelcost}}(S)$ is the total travel time for a vehicle to complete route S, including the waiting time if the vehicle arrives at a location early. The penalty function of route S, $f_{\text{penalty}}(S)$, defines the punishments for violating some of the constraints. The vehicle receives a penalty if it is overloaded or late at any location. In a route, if any specific task i violates the precedence constraint, an adjustment procedure will be performed. This procedure swaps the positions of location i^+ and i^- and makes the pickup location i^+ to appear before the delivery location i^-.

A route may violate the constraints during the exploration of genetic algorithm. A route is *feasible* if it does not violate any constraint; otherwise, it is *infeasible*. According to the definition of $\Phi(S)$, the value of $\Phi(S_{\text{infeasible}})$ is typically much larger than that of

$\Phi(S_{\text{feasible}})$, where S_{feasible} denotes a feasible route and $S_{\text{infeasible}}$ is an infeasible route. The goal of the 1-PDPTW is to find a feasible route S such that minimize $\Phi(S)$.

3.3 Crossover

In this paper, we consider four crossover operators. The order crossover [34] (OX) and uniform order-based crossover [32] (UOX) are two popular traditional crossovers to solve routing-related problems. Merge crossover #1 (MX1) and merge crossover #2 (MX2) are recently invented by Blanton [40] for solving VRPTW; these two merge crossovers utilize global knowledge to explore their search space. The two traditional crossover operators have been introduced by many exhaustive studies [32,34].

Most traditional order-based crossover operators do not have strong connections to the constraints in the problem domain. In contrast, the merge crossover operators utilize global knowledge about 1-PDPTW constraints, in the form of a *global precedence vector*, in order to achieve the precedence relationship among the genes. For example, in 1-PDPTW, the order of time windows can be represented as a global precedence vector for the merge crossovers. Operators MX1 and MX2 will produce new chromosomes according to the precedence among the genes. Operator MX1 produces a child with the sequence of order close to the global precedence vector; while MX2 produces an offspring in which the genes with lower priority are moved to the end of the chromosome.

3.4 Mutation

A mutation operator works on a single chromosome, which will be substituted by the mutated individual. In this approach, instead of using a fixed mutation rate, mutation is applied only when the offspring are identical to their parents.

The genetic algorithms in our experiments applied two mutation operators in solving the 1-PDPTW. The first mutation operator, named 2-point mutation, selects two genes randomly, and their positions are interchanged. This operator creates a new route with four different edges from its original route. The second mutation operator chooses two cut sites randomly and reverse the sub-route among the cut sites. This mutation operator is identical to the 2-opt move in TSP [32], in which the new route differs from the original one by two edges.

4. FAMILY COMPETITION GA

The family competition genetic algorithm (FCGA) is a modern approach introduced by Yang [44]. Adapting the concept of families to traditional GA yields the principal of FCGA. Once an individual has been selected to perform crossover, traditional GA selects another individual to produce a single offspring, whereas FCGA produces a family with more than one offspring, one from each randomly selected mate. In FCGA, the scheme of family competition maintains a constant size of the population, that is, only the champion of each family survives.

Algorithm 1 The procedures of FCGA

1: $t = 0$;
2: Initial population: $P^t \leftarrow \{I_1^t, I_2^t, \ldots, I_m^t\}$;
3: **Evaluation**: $\Phi(I_1^t), \Phi(I_2^t), \ldots, \Phi(I_m^t)$
4: **Repeat**
5: T $\leftarrow \emptyset$;
6: **for** $i = 1$ **to** m **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: **Recombination**: $c_j \leftarrow O_m(O_c(F_i^t, A_i^t))$;
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: $P^{t+1} \subset \{P^t \cup T\}$;
18: $t = t + 1$;
19: **until** reach the termination condition
20: Output the solutions;

Algorithm 1 elaborates the procedure of the family competition genetic algorithm. The first three steps are the initialization process of FCGA. For any given generation t, the population P^t contains m individuals. The evaluation function Φ is as defined in Eq. (1) of Section 3.2.

Steps 4 through 19 present the main procedure of FCGA in generation t. Each individual I_i^t in P^t, where $i = 1, 2, \cdots, m$, takes turn to be the father F_i^t in creating a family C_i^t of u offspring. The family construction process is detailed in Steps 9 to 14. To produce a new offspring c_j, an alternative parent A_j^t, which should be distinct from the family father F_i^t, is randomly selected from population P^t. Crossover operator O_c and mutation operator O_m are then applied to F_i^t and A_j^t. In Step 15, function $best(C_i^t)$ returns the individual with the best, *i.e.*, lowest, fitness value from family C_i^t. The best members of each family are collected in a temporary set T. In Step 17, the best m individuals from either P^t or T are selected to form the next generation. Algorithm 1 repeats the main loop until

the termination conditions are met. Upon termination, FCGA outputs the best solutions explored so far.

5. EXPERIMENTAL DESIGN AND RESULTS

A major problem for research on solving PDPTW is the lack of standard test sets. While standard data sets for several related problems do exist, they cannot be used to test PDPTW. The TSP data sets do not include precedence, capacity or time window constraints; while the VRPTW data sets lack precedence constraints. In order to compare the genetic algorithms for solving PDPTW under various combinations of operators and parameters, it is necessary to create our own test data sets.

5.1 Experimental Design

Algorithm 2 describes the procedure for generating the random test sets used in our experiments. The main challenge is to create test data that is random enough to provide good coverage on the problem space, while ensuring the constraints are not too tight to exclude all possible solutions.

Algorithm 2 Create test data for 1-PDPTW

Require: Given the number of task n, and the width of time window *width*

Ensure: To produce a task set N with n transportation requests, including pickup and delivery locations from a set V, loads $\{q_1 \cdots q_n\}$ and time window $\{[a_1,b_1] \cdots [a_n, b_n]\}$, which admits feasible solutions.

1: 0 is an initial depot;
2: $N = \{1, \cdots, n\}$
3: **for** $i \in N$ **do**
4: Generate the pickup location $i^+ \in V^+$ randomly;
5: Generate the delivery location $i^- \in V^-$ randomly;
6: Generate the load q_i randomly;
7: **end for**
8: $V = \{0 \cup V^+ \cup V^-\}$;
9: Evaluate the traveling time d_{rs}, $r, s \in V$;
10: Randomly generate a route, which satisfies the precedence contraint;
11: $AverageTime = \dfrac{\sum_{r \in V} \sum_{s \in V} d_{rs}}{|V| \times |V|}$
12: Suppose that the vehicle arrives location i at time t_i;
13: **for** $i \in V^+ \cup V^-$ **do**
14: $a_i = t_i - random(width) \times AverageTime$;
15: $b_i = t_i + random(width) \times AverageTime$;
16: **end for**
17: $[a_i, b_i]$ is the time window of location $i \in V$;
18: Output the test data;

Each generated test set consists of n tasks with $2n + 1$ locations. Steps 1 to 8 in Algorithm 2 initialize the essential elements of tasks, including the pickup locations, delivery locations and the loads. Step 9 calculates the travel time between any locations involved in the requests.

Moreover, this algorithm guarantees the existence of solutions. To assure the production of a test set with at least one solution, Algorithms 2 randomly generates a primary route that satisfies the intrinsic precedence constraints. The arrival time t_i of location i is used to generate the corresponding time window $[a_i,b_i]$ using the parameter *width* in deciding the width of time window. Function *random*(*width*) in Steps 14 and 15, randomly generates a real number between 0 and *width*. As a result, the difference between a_i and b_i is no more than $2 \times width \times AverageTime$.

5.2 Experimental Results

In our experiments, the family competition genetic algorithm (FCGA) is compared with the traditional genetic algorithm (GA) using four crossovers operators under three different crossover rates (0.45, 0.60, and 0.75). Comparative results are tabulated for the four crossover operators, which are referred to as order-based crossover (OX), uniform crossover (UOX) [32], merge crossover #1 (MX1), and merge crossover #2 (MX2) [40]. The experiments also adopt two mutation operators: 2-point mutation and the 2-opt mutation. For each test case, we perform 30 trials under the same configuration of parameters.

Table 1 shows the best results of running the four crossover operators respectively. The first column represents the number of transportation requests. Under the common heading "Genetic Operators & Approaches," Columns 2 to 9 list the results of using the four crossover operators. Each cell consists of two elements: the top value indicates the best result found by the corresponding approach, whereas the value in the bottom represents the number of times when the best result is found. The latter is also called the *appearance count*. Symbol 'x' indicates that no feasible solution can be found. Values equal to the best-known result are printed in italicized style. The rightmost columns record the best solutions obtained by using GA, FCGA, and dynamic programming (DP) [9] respectively.

The results in Table 1 show that the order-based crossover OX is unable to find any feasible solution for relatively small problems. Therefore, it is not suitable for 1-PDPTW. On the other hand, dynamic programming produces optimal solutions for problems with size smaller than 50; the uniform crossover and two merge crossover operators can find optimal solutions in

Table 1 A comparison of the best results

Task Size	Genetic operators & approaches								The best or optimal result		
	OX		UOX		MX1		MX2				
	GA	FCGA	GA	FCGA	GA	FCGA	GA	FCGA	GA	FCGA	DP
10	875	*872*	*872*	*872*	*872*	*872*	*872*	*872*	*872*	*872*	*872*
	1	*2*	*18*	*27*	*30*	*30*	*30*	*30*			
20	x	2186	*2030*	*2030*	*2030*	*2030*	*2030*	*2030*	*2030*	*2030*	*2030*
	x	1	*22*	*30*	*1*	*1*	*21*	*11*			
30	x	x	*3713*	*3713*	*3713*	3720	*3713*	*3713*	*3713*	*3713*	*3713*
	x	x	*5*	*20*	*2*	4	*24*	*7*			
40	x	x	*4386*	*4386*	4389	4389	*4386*	*4386*	*4386*	*4386*	*4386*
	x	x	*3*	*23*	2	10	*28*	*12*			
50	x	x	*5752*	*5752*	5753	5723	*5752*	5753	*5752*	*5752*	*5752*
	x	x	*1*	*17*	30	30	*3*	24			
60	x	x	*5658*	*5658*	*5658*	*5658*	*5658*	*5658*	*5658*	*5658*	x
	x	x	*3*	*5*	*3*	*1*	*29*	*30*			
70	x	x	*7221*	*7221*	7252	7239	*7221*	*7221*	*7221*	*7221*	x
	x	x	*2*	*19*	10	1	*2*	*2*			
80	x	x	*7849*	*7849*	7903	7909	7850	*7849*	*7849*	*7849*	x
	x	x	*1*	*4*	1	18	1	*1*			
90	x	x	*x*	*8618*	8622	8645	*8618*	*8618*	*8618*	*8618*	x
	x	x	*x*	*5*	1	1	*1*	*1*			
100	x	x	*10600*	*10600*	10616	*10600*	*10600*	*10600*	*10600*	*10600*	x
	x	x	*1*	*5*	4	*3*	*10*	*26*			

most cases. The results also show that FCGA consistently improves the quality of solutions and the chance of producing optimal solutions using the uniform crossover operator. When applied with the merge crossover operators, FCGA improves the quality of solutions for most, while slightly worsens the solutions of MX1 for task size 30, 80 and 90, as well as the solution of MX2 for 50 tasks.

Consider the chance of producing the best solutions for the various combinations. Applying UOX in FCGA generally increases the appearance count of the best results. In contrast, the appearance count decreases for MX1 in FCGA for task size 60 and larger; FCGA with MX2 obtains fewer optimal solutions for task size of 20, 30 and 40.

To evaluate the quality of the solutions found by the various genetic algorithms, Fig. 1 depicts the best and average fitness values of the solutions for 1-PDPTW involving 80 tasks. The x-coordinate denotes the number of generations, whereas the y-coordinate correspond to the fitness values. Compared with FCGA, the average fitness values for GA are much higher (*i.e.*, worse) in every generation and the curve converges more slowly. In addition, the results indicate that the best results obtained by FCGA and GA are very close and the lines converge at the same point.

Most of the configurations in our experiments generated similar results to the ones shown in Fig. 1. There are notable exceptions. For example, Fig. 2 shows the best and average fitness values of solutions found by MX2 over thirty trials on 50-task PDPTWs. The curve denoting the average fitness values for FCGA converges prematurely into a higher value. One possible explanation is that the merge crossover operators can create near-optimal solutions earlier in the evolution process based on some domain knowledge. Unfortunately, FCGA has a higher tendency of being

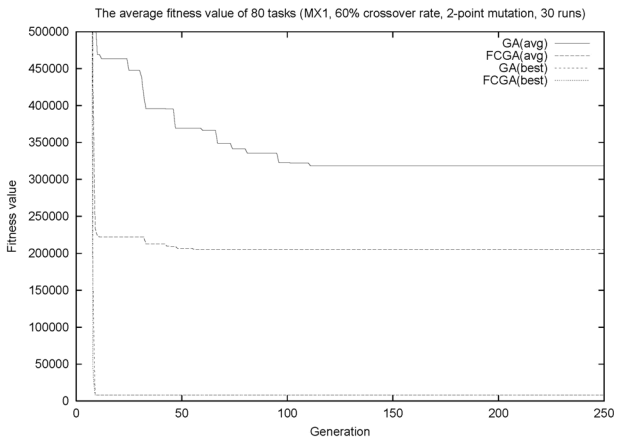

Fig. 1 The best and average fitness values of MX1 in 80 tasks

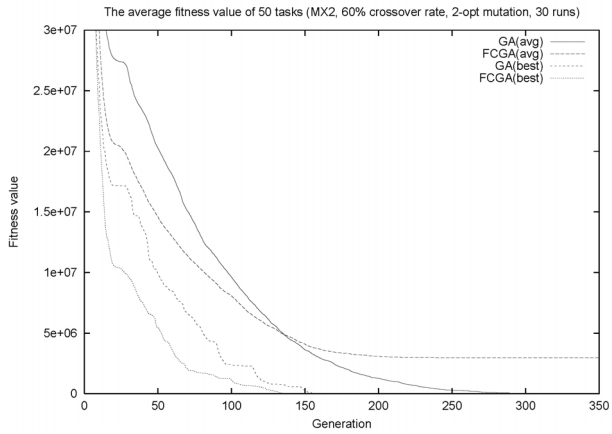

Fig. 2 The best and average fitness values of MX2 in 50 tasks

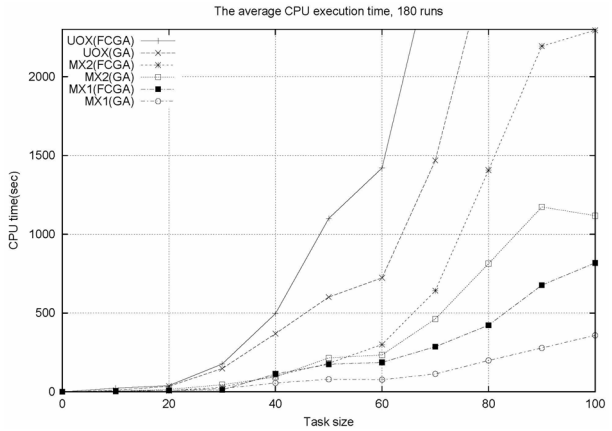

Fig. 3 A comparison of CPU time (UOX, MX1, and MX2)

trapped in a local minimum, and converges to a non-optimal solution.

Figure 3 illustrates the average CPU time required by FCGA and GA in solving the problems. The x-coordinate represents the problem size, and the y-coordinate denotes the execution time measured in seconds. In general, the execution time for GA is shorter than that of FCGA due to the overhead in the reproductive process. Figure 3 also show that UOX takes more time than the merge crossover operators in finding a solution. The experimental results in Table 1 and Fig. 3 demonstrate that UOX works well with FCGA in generating the best solutions despite a longer execution time.

In real-world applications, it is often better to start serving customer requests based on a feasible (or near-optimal) solution instead of keeping them waiting while searching for the optimal solution. Table 2 summarizes the hit ratio, in terms of percentage, of finding a feasible solutions. The results indicate that FCGA improves the probability of obtaining feasible solutions in most cases. Based on the results in Fig. 3 and Table 2, we can conclude that MX1 and MX2 are the better choices for real-time applications.

Table 2 Hit ratios of feasible solution in 180 trials

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

Let us summarize the results from our experiments. First, FCGA improves the solution quality of UOX, both the solution costs and the appearance count. Second, for uniform order-based crossover operator like UOX, FCGA can enhance its local search capability, thereby increase the opportunity of reaching the optimal solution. On the other hand, while FCGA improves the execution time of the merge crossover operators, it does not increase the solution qualities as it tends to get stuck in the local minimum.

6. CONCLUSION AND DISCUSSION

This paper presented a comprehensive study on genetic algorithm-based approach to solving the single-vehicle pickup and delivery problem with time constraints. Genetic algorithms have been shown to find feasible or near-optimal solutions when traditional methods would fail within a reasonable amount of time. Instead of utilizing offsprings of parent chromosomes directly, the Family Competition Genetic Algorithm (FCGA) selects a champion offspring from siblings from the same parents to maintain diversity and balance of population. By incorporating the concept of families, FCGA further improves solution quality and increases the probability of finding the optimal solutions without much extra resource demands.

Extensive experiments were conducted to compare the FCGA with traditional GA's under various crossover and mutation operators. The results showed that the FCGA succeeds in finding the optimal solutions for 1-PDPTW under 50 tasks, and can obtain near-optimal or feasible solutions in most problems up to 100 tasks.

Additionally, by comparing the solution cost of FCGA with that of traditional GA, we found that FCGA improves the solution qualities in most cases.

One important observation from the experiments in solving the 1-PDPTW is that not all recombination operators benefit from FCGA. In particular, FCGA is best suited for genetic operators that explore the search space uniformly, such as uniform crossover. For genetic operators designed to do greedy search, FCGA may be trapped in local minimum. Our experiments showed that applying family competition in merge crossovers did not improve the solutions substantially. We expect to continue designing additional recombination operators in order to further understand the characteristics of the proposed scheme in solving PDPTW and other related problems.

REFERENCES

[1] O. B. G. Madsen, K. Tosti and J. Væds, "A heuristic method for dispatching repair men," *Annals of Operations Research*, Vol. 61, 1995, pp. 213–226.

[2] Y. Shen, J. Y. Potvin, J. M. Rousseau and S. Roy, "A computer assistant for vehicle dispatching with learning capabilities," *Annals of Operations Research*, Vol. 61, 1995, pp. 189–211.

[3] R. W. Eglese, "Routeing winter gritting vehicles," *Discrete Applied Mathematics*, Vol. 48, 1994, pp. 231–244.

[4] Y. O. Li and R. W. Eglese, "An interactive algorithm for vehicle routeing for winter-gritting," *Journal of the Operational Research Society*, Vol. 47, 1996, pp. 217–228.

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

[6] J. K. Lenstra and A. H. G. Rinnooy Kan, "Complexity of vehicle routing and scheduling problems, *Networks*, Vo1. 11, 1981, pp. 221–227.

[7] H. N. Psaraftis, "A dynamic programming solution to the single vehicle many-to-many immediate request dial-a-ride problem," *Transportation Science*, Vol. 14, No. 2, 1980, pp. 130–154.

[8] H. N. Psaraftis, "An exact algorithm for the single vehicle many-to-many dial-a-ride problem with time windows," *Transportation Science*, Vol. 17, No. 3, 1983, pp. 351–357.

[9] H. N. Psaraftis, "Scheduling large-scale advance-request dial-a-ride systems," *American Journal of Mathematical and Management Sciences*, Vol. 6, No. 3 & 4, 1986, pp. 327–341.

[10] J. Desrosiers, Y. Dumas and F. Soumis, "A dynamic programming solution of the large-scale single-vehicle dial-a-ride problem with time windows," *American Journal of Mathematical and Management Sciences*, Vol. 6, No. 3 & 4, 1986, pp. 301–325.

[11] T. R. Sexton and Y. M. Choi, "Pickup and delivery of partial loads with soft time windows," *American Journal of Mathematical and Management Sciences*, Vol. 6, No. 3 & 4, 1986, pp. 369–398.

[12] Y. Dumas, J. Desrosiers and F. Soumis, "The pickup and delivery problem with time windows," *European Journal of Operational Research*, Vol. 54, 1991, pp. 7–22.

[13] M. W. P. Savelsbergh and M. Sol, "The general pickup and delivery problem," *Transportation Science*, Vol. 29, No. 1, 1995, pp. 17–29.

[14] M. Sigurd, D. Pisinger and M. Sig, "The pickup and delivery problem with time windows and precedences," Technical report, University of Copenhagen, August 2000.

[15] O. B. G. Madsen, H. G. Ravn and J. M. Rygaard, "A heuristic algorithm for a dial-a-ride problem with time windows, multiple capacities and multiple objective," *Annals of Operations Research*, Vol. 60, 1995, pp. 193–208.

[16] M. Charikar and B. Raghavachari, "The finite capacity dial-a-ride problem," *IEEE Symposium on Foundations of Computer Science*, 1998, pp. 458–467.

[17] N. Christofides, "Vehicle routing," E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan and D. B. Shmoys, Eds., *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, Elsevier Science Publishers, John Wiley & Sons, Ltd., 1985, pp. 431–448.

[18] G. Laporte, "The traveling salesman problem: an overview of exact and approximate algorithms," *Management Science*, Vol. 59, 1992, pp. 231–247.

[19] S. Anily and G. Mosheiov, "The traveling salesman problem with delivery and backhauls," *Operations Research Letters*, Vol. 16, 1994, pp. 11–18.

[20] M. Gendreau, G. Laporte and D. Vigo, "Heuristics for the traveling salesman problem with pickup and delivery," *Computers and Operations Research*, Vol. 26, No. 7, 1999, pp. 699–714.

[21] G. Choi, C. Moon, J. Kim and Y. Seo, "An

efficient genetic algorithm for the traveling salesman problem with precedence constraints," *European Journal of Operational Research*, Vol. 140, No. 3, 2002, pp. 606–617.

[22] H. N. Psaraftis, "Dynamic vehicle routing problems," B. L. Golden and A. A. Assad, Eds., *Vehicle Routing, Methods and Studies*, Elsevier Science Publishers, North-Holland, Amsterdan, 1988, pp. 223–248.

[23] G. Laporte, "The vehicle routing problem: an overview of exact and approximate algorithms," *Management Science*, Vol. 59, 1992, pp. 345–358.

[24] B. L. Golden and A. A. Assad, "Vehicle routing with time-window constraints," *American Journal of Mathematical and Management Sciences*, Vol. 6, No. 3 & 4, 1986, pp. 251–260.

[25] M. M. Solomon and J. Desrosiers, "Time window constrained routing and scheduling problems," *Transportation Science*, Vol. 1, No. 1, 1988, pp. 1–13.

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

[27] G. Desaulniers, J. Desrosiers, A. Erdmann, M. M. Solomon and F. Soumis, "The VRP with pickup and delivery," Cahiers du GERARD G-2000-25, Ecole des Hautes Etudes Commerciales, Montreal, 2000.

[28] G. Laporte, "Routing problems: a bibliography," *Annals of Operations Research*, Vol. 61, 1995, pp. 227–262.

[29] J. Grefenstette, R. Gopal, B. Rosmaita and D. Gucht, "Genetic algorithms for the traveling salesman problem," *Proceedings of the First International Conference on Genetic Algorithms and Their Applications*, 1985, pp. 160–168.

[30] D. Whitley, T. Starkweather and D. Fuquay, "Scheduling problems and traveling salesman: the genetic edge recombination operator," *Proceedings of the Third International Conference on Genetic Algorithms and Their Applications*, 1989, pp. 133–140.

[31] T. Starkweather, S. McDaniel, K. Mathias, D. Whitley and C. Whitley, "A comparison of genetic sequencing operators," *Proceedings of the Fourth International Conference on Genetic Algorithms and Their Applications*, 1991, pp. 69–76.

[32] G. Syswerda, "Schedule optimization using genetic algorithms," L. Davis, Ed., *Handbook of Genetic Algorithms*, van Norstrand Reinhold, New York, 1991, pp. 332–349.

[33] D. E. Goldberg and R. Lingle, Jr., "Allels, loci and the traveling salesman problem," *Proceedings of the First International Conference on Genetic Algorithms and Their Applications*, 1985, pp. 154–159.

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

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

[36] B. Freisleben and P. Merz, "New genetic local search operators for the traveling salesman problem," *Proceedings of 1996 IEEE International Conference on Evolutionary Computation*, 1996, pp. 616–621.

[37] B. Freisleben and P. Merz, "New genetic local search operators for the traveling salesman problem," *Proceedings of the 4th International Conference on Parallel Problem Solving from Nature (PPSN'96)*, 1996, pp. 890–899.

[38] J. Y. Potvin, "Genetic algorithms for the traveling salesman problem," *Annals of Operations Research*, Vol. 63, 1996, pp. 339–370.

[39] P. Larranaga, C. Kuijpers, R. Murga, I. Inza and S. Dizdarevich, "Genetic algorithms for the travelling salesman problem: a review of representations and operators," *Artificial Intelligence Review*, Vol. 13, 1999, pp. 129–170.

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

[41] W. R. Jih, Y. P. Chen and Y. J. Hsu, "A comparative study of genetic algorithms for vehicle routing with time constraints," *Proceedings of the 1996 International Computer Symposium*, 1996, pp. 17–24.

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

[43] H. K. Tsai, J. M. Yang and C. Y. Kao, "A genetic

algorithm for traveling salesman problems," *Proceedings of the Genetic and Evolutionary Computation Conference* (*GECCO-2001*), 2001, pp. 687–693.

[44] J. M. Yang and C. Y. Kao, "Integrating adaptive mutations and family competition into genetic algorithms as function optimizer," *Soft Computing*, Vol. 4, No. 2, 2000, pp. 89–102.

Wan-rong Jih (紀 婉 容) is a Ph.D. candidate in the Department of Computer Science and Information Engineering at the National Taiwan University, and her current research focuses on the optimization searching algorithms, vehicle routing problems, and protein folding simulation.

Jane Yung-jen Hsu (許 永 真) received her Ph.D. in Computer Science from Stanford University in 1991. She is an associate professor of Computer Science and Information Engineering at National Taiwan University. Her research interests include intelligent agents, data mining, mobile robots, and e-commerce technologies.

Professor Hsu serves on the editorial board of Intelligent Data Analysis—An International Journal and International Journal of Computational Intelligence. She is actively involved in many key international conferences, e.g. as the Program co-Chair for the 2005 IEEE International conference on e-Technology, e-Commerce, and e-Service. She is a member of AAAI, IEEE, ACM, and the Phi Tau Phi Scholastic Honor Society, an executive committee member of TAAI, and helps initiate the Asian-Pacific Chapter of the IEEE Technical Committee on E-Commerce.

收稿日期 92 年 12 月 17 日、修訂日期 93 年 2 月 18 日、接受日期 93 年 2 月 20 日
Manuscript received December 17, 2003, revised February 18, 2004, accepted February 20, 2004