

Dynamic Vehicle Routing Using Hybrid Genetic Algorithms

Wan-rong Jih
jih@robot.csie.ntu.edu.tw

Jane Yung-jen Hsu
yjhsu@csie.ntu.edu.tw

Department of Computer Science and Information Engineering
National Taiwan University
Taipei, Taiwan

Abstract

This paper presents a novel approach to solving the single-vehicle pickup and delivery problem with time windows and capacity constraints (or single-vehicle PDPTW). While dynamic programming has been used to find the optimal routing to a given problem, it requires time exponential in the number of tasks. Therefore, it often fails to find the solutions under real-time conditions in an automated factory. This research explores anytime problem solving using genetic algorithms. By utilizing optimal but possibly partial solutions from dynamic programming, the hybrid genetic algorithms can produce near-optimal solutions for problems of sizes up to 25 percent bigger than what can be solved previously. This paper reports the experimental results of the proposed hybrid approach with four different crossover operators as well as three mutation operators. The experiments demonstrated the advantages of the hybrid approach with respect to dynamic task requests.

1 Introduction

To this day, only relatively small VRP instances can be solved to optimality.

In this paper, we consider the single-vehicle pickup and delivery problem with time windows and capacity constraints (single-vehicle PDPTW), which will be formally defined in section 2. Most exact algorithms capable of finding optimal solutions to this problem are based on dynamic programming. For a single-vehicle PDPTW with n tasks, the computational complexity of dynamic programming has been shown to be bounded by $(2n + 1)^2 3^n$ [5]. Therefore, existing algorithms cannot satisfy real-time requests.

In order to handle dynamic task requests efficiently, it is desirable to have heuristic algorithms that can generate (sub-optimal) solutions to large problems on

demand. The solutions should be improved incrementally whenever more time becomes available. Genetic algorithms (GAs) can generate incrementally better solutions at each succeeding generation regardless of the problem size. Therefore, they provide a reasonable solution to the dynamic single-vehicle PDPTW problem. Unfortunately, the solutions from simple GAs are often of poor quality.

This research explores hybridizing genetic algorithms with dynamic programming in order to take advantage of the complementary properties of both in solving the dynamic single-vehicle PDPTW problem. The proposed hybrid approach starts by using dynamic programming to generate the optimal routes. If optimal solutions are not found within the specified time slot, the partially constructed routes are passed to the genetic algorithms. The sub-routes provide the basis for generating an initial population that often leads to better convergence than a randomly generated initial population. The hybrid approach enables dynamic programming to achieve real-time performance, and improves genetic algorithms in approximating near-optimal solutions.

The next section introduces the single-vehicle PDPTW, and section 3 presents the hybrid approach. The experimental results and analysis are shown in section 4, followed by the conclusion in section 5.

2 The Single-vehicle PDPTW

We should know the static single-vehicle PDPTW before we focus our attention on the dynamic single-vehicle PDPTW. The section 2.1 will describe the static case of single-vehicle PDPTW. The dynamic case will be presented in section 2.2.

2.1 Static case

Suppose that $N = \{1, \dots, n\}$ is a set of tasks, where n is the number of tasks. To accomplish each task

$i \in N$, the vehicle should pick up the goods at the pickup location i^+ and then transport the goods to the delivery location i^- . According to these pickup and delivery locations, $V^+ = \{i^+ \mid i \in N\}$ and $V^- = \{i^- \mid i \in N\}$ indicate the set of pickup locations and the set of delivery locations respectively. Let $G = (V, A)$ be a graph where $V = \{0\} \cup V^+ \cup V^-$ is a set of vertices and $A = \{(r, s) \mid r \neq s, r, s \in V\}$ is a set of arcs. Vertex 0 represents an initial depot of the vehicle. Each arc $(r, s) \in A$ is associated with a non-negative travel time d_{rs} , and each task $i \in N$ is associated with a non-negative demand q_i . The total demand of a vehicle route may not exceed the vehicle capacity Q . Specifically, for every task i are required for picking up the goods within the time interval $[a_{i^+}, b_{i^+}]$, and transporting the goods to its delivery location i^- on the time interval $[a_{i^-}, b_{i^-}]$. In addition, let t_r and l_r represent the arrival time and the load of a vehicle respectively, when the vehicle arrives at a location $r \in V$.

Our goal is to find a vehicle route starting from an initial depot, finishing all the requests of tasks, and ending at one of the delivery locations. According to the route, both the total traveling time and waiting time of the vehicle are minimized. Certainly, this route must be a *feasible route* that is satisfied the capacity and the time windows constraints of each task. In addition, the feasible routes visit the pickup location i^+ before the delivery location i^- of task $i \in N$.

Initially, we assume that the vehicle is stopped at an initial depot 0. Either the pickup location or the delivery location are associated with an advanced time window $[a_r, b_r]$, where $r \in V^+ \cup V^-$. The weight of goods q_i for a task $i \in N$ is known, when the task i is assigned. We assume the value of travel time d_{rs} is already known, where $r, s \in V$. If the vehicle arrives at a location $r \in V^+ \cup V^-$ and its arrival time t_r is earlier than a_r , it has to wait. Every task $i \in N$ is subject to $a_{i^+} \leq b_{i^+}$, $a_{i^-} \leq b_{i^-}$ and $b_{i^-} - a_{i^+} \geq d_{i^+i^-}$.

The solution cost of the static single-vehicle PDPTW problem is minimized.

$$Z_{objective} = \omega_1 \sum_{r,s \in V} d_{rs} x_{rs} + \omega_2 \sum_{r \in V^+ \cup V^-} f_{waiting}(r) \quad (1)$$

The notation x_{rs} , $r, s \in V$, is defined as

$$x_{rs} = \begin{cases} 1, & \text{if arc } (r, s) \text{ is used by vehicle;} \\ 0, & \text{otherwise.} \end{cases}$$

The function $f_{waiting}(r)$, defines the waiting time of a location $r \in V^+ \cup V^-$ as

$$f_{waiting}(r) = \begin{cases} a_r - t_r, & \text{if vehicle arrives at location } r \text{ early;} \\ 0, & \text{otherwise.} \end{cases}$$

The first operand of the equation (1) is the total traveling time needed by the vehicle for completing the route, and the second operand of equation (1) is the total waiting time. The two factors ω_1 and ω_2 are the weights, which reflect the relative importance of these two parts. We generally set $\omega_1 + \omega_2 > 0$ and $\omega_1, \omega_2 \geq 0$.

Solutions to the single-vehicle PDPTW are subject to the following constraints[2]. First, the vehicle starts from an initial depot, and each location is visited only once. Second, each task $r \in N$ should be picked up before it is delivered, which is called *precedence constraints*. Third, the total load allocated to a vehicle cannot exceed its capacity. This constraint is named *capacity constraints*. Fourth, whenever the vehicle arrives at location $i \in V^+ \cup V^-$ at time t_i , the criterion $t_i \leq b_i$ must be satisfied. In general, we named the fourth constraint as *time window constraints*. Finally, the vehicle routes will be open paths, ending at any one of the delivery locations.

2.2 Dynamic case

In the static cases, tasks to the problem do not change, either during the execution of the algorithm, or during the eventual execution of the route. By contrast, in dynamic cases, tasks may change during the execution of the algorithm and the eventual execution of the route. For the requirements of a dynamic vehicle routing problem, we should modify the objective function $Z_{objective}$ to accomplish a route in real time. A modified objective function is defined as

$$Z_{dynamic} = Z_{objective} + [\alpha_1 \sum_{r \in V^+ \cup V^-} f_{delay}(r) + \alpha_2 \sum_{r \in V^+ \cup V^-} f_{overload}(r)]$$

The objective function $Z_{dynamic}$ is based on the objective function of the static case, but with some penalties. For any $r \in V^+ \cup V^-$, the function $f_{delay}(r)$ and $f_{overload}(r)$ represent the vehicle delay time and overload at location r respectively, where α_1 and α_2 are penalty coefficients.

The functions $f_{delay}(r)$ and $f_{overload}(r)$ are defined as following ($r \in V^+ \cup V^-$):

$$f_{delay}(r) = \begin{cases} t_r - b_r, & \text{if a vehicle arrives at location } r \text{ lately;} \\ 0, & \text{otherwise.} \end{cases}$$

$$f_{overload}(r) = \begin{cases} l_r - Q, & \text{if the current load } l_r \text{ exceeds the vehicle capacity } Q; \\ Q; \\ 0, & \text{otherwise.} \end{cases}$$

According to the definition of $Z_{dynamic}$, we will try to minimize it.

3 Hybrid Approach

Figure 1 shows the architecture of the hybrid genetic algorithms.

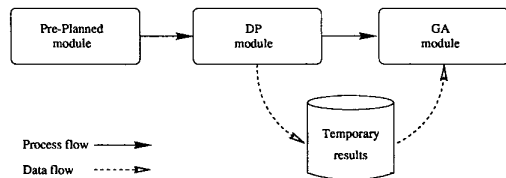


Figure 1: Architecture of hybrid GA

3.1 Pre-planned module

A pre-planned module will arrange the tasks and prepare the information for the DP module. The pre-planned module collects the tasks that have not been finished, and set a new initial depot of the vehicle. The execution time and the load of vehicle are not equal to zero and the pre-planned module should maintain the proper values of them.

3.2 DP module

The DP module performs a dynamic programming algorithm. When a specific time is expired, the DP module will pass its unfinished sub-routes to a temporary result pool; these unfinished sub-routes will be the initial population of genetic algorithms. We adopt the dynamic programming approach that proposed by Psaraftis[4]. In the current state of knowledge, the single-vehicle dial-a-ride problems can rarely be achieved to optimization when the number of tasks is more than 40. As to the dial-a-ride problems, they might be a special case of PDPTW when every task demands of the PDPTW are equal. Detailed algorithms of dynamic programming can be found in many publications[4, 5, 3].

3.3 Genetic algorithm

We summarize the genetic algorithm used in our work as Algorithm 1. The procedure of step2 will generate an initial population. The initial population of the hybrid genetic algorithms is based on the partially constructed routes of DP module; the sub-routes are in front of the initial routes. According to the step 7, we apply the *tournament selection* to select two parents;

Algorithm 1 Genetic algorithm of single-vehicle PDPTW

```

1:  $t = 0$ ;
2: Generate a population  $P(t)$ ;
3: Use  $Z_{dynamic}$  to evaluate chromosomes in  $P(t)$ ;
4: while termination condition not satisfied do
5:   Initialize a temporary population  $P'$ ;
6:   for  $i = 1$  to  $|P(t)|$  do
7:     Select two parents from  $P(t)$ ;
8:     Perform crossover;
9:     if the offspring and its parents are identical then
10:       Perform mutation;
11:     end if
12:     Place the offspring into  $P'$ ;
13:   end for
14:    $t = t + 1$ ;
15:   Replace  $P(t)$  by  $P'$ ;
16: end while
17: Output the solutions;

```

and the tournament size is 2. Four crossover operators and three mutation operators will be discussed in this paper. The algorithm will be executed until one of the following conditions is met: (1) The number of generation is larger than the maximum generation. (2) The solution that has the best fitness value appears more than a specific sequence. (3) $\frac{\sigma(P)}{m(P)} \leq 0.005$. The $\sigma(P)$ and $m(P)$ are the standard deviation and the mean of the fitness value in the population P respectively. Currently, we set the values of maximum generation and the appearance sequence are equal to 2000 and 100 respectively.

3.3.1 Representation

Given a single-vehicle PDPTW with n tasks, a solution is encoded as a chromosome, which is represented by a permutation of locations from 1^+ to n^- . For example, in 2 tasks problem, if a route is $0 \rightarrow 1^+ \rightarrow 2^+ \rightarrow 2^- \rightarrow 1^-$, the representation of the chromosome will be $(1^+ 2^+ 2^- 1^-)$. This representation could be understood easily and the space requirement is less than other representations. In addition, it is convenient for implementation.

In the single-vehicle PDPTW, the precedence constraints can not be violated. The representation does not preclude routes from violating the precedence constraints. A simple algorithm is used to maintain the feasibility of the corresponding routes. A chromosome adjustment algorithm will exchange the positions of i^+ and i^- , when the vehicle visits the delivery location i^- before the pickup location i^+ of a task $i \in N$. According to the adjustment algorithm, the chromo-

somes will be satisfied the precedence constraints by making the pickup location be appeared before the delivery location for any specific task.

3.3.2 Crossover operators

In the present work, we compare to the performance of four crossover operators on the single-vehicle PDPTW problem. These operators are order crossover (OX), uniform order-based crossover (UOX), merge cross #1 (MX1) and merge cross #2 (MX2)[1]. The first two operators are traditional crossover operators and the last two operators use a global precedence vector to be the guidance of crossover. We will not describe the two traditional crossover operators. The other two crossover operators are described below.

Most traditional crossover operators for order-based GAs do not have strong connection to the constraints of the problems they are applied. If we apply the traditional crossover operators to the single-vehicle PDPTW, they might not be conducive to the searching process of optimal solutions because the information of the constraints is not used by these operators. On the other hand, the notion of merge crossover operators is that a global precedence among genes independent of any chromosome, rather than the behavior of traditional crossover operators that defines a local precedence among genes specific to a chromosome. That is, each gene in the chromosome has a precedence relationship 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 our single-vehicle PDPTW, each gene is either a delivery point or a pickup point and has an associated time window. Therefore, we can make use of the time window $[a_r, b_r]$ of the location $r \in V^+ \cup V^-$ and a precedence relationship among each earliest processing time a_r . Given $r, s \in V^+ \cup V^-$, if a_r less than a_s , let the location r be appeared before the location s in the global precedence vector because it will be a reasonable solution to serve the location r before the location s . In other words, we can sort all a_r in the ascending order, and the sorting result will be a global precedence vector.

We demonstrate the operations of MX1 with an example. Suppose that the set of tasks $N = \{1, 2, 3, 4\}$ and the set of locations $V = \{0, 1^+, 1^-, 2^+, 2^-, 3^+, 3^-, 4^+, 4^-\}$. Given a global precedence vector $(1^+, 2^+, 3^+, 4^+, 1^-, 2^-, 3^-, 4^-)$, the location 1^+ is top-priority, and the location 2^- takes precedence over the locations 3^- and 4^- . In other words, the location 1^+ has the precedence of the oth-

ers and the location 4^- has the lowest precedence. The MX1 operator on two chromosomes (say P_1 and P_2) produces single offspring, as shown below:

$$\begin{array}{l} P_1: 1^+ \ 1^- \ 2^+ \ 2^- \ 3^+ \ 3^- \ 4^+ \ 4^- \\ P_2: 2^+ \ 3^+ \ 1^+ \ 2^- \ 1^- \ 4^+ \ 4^- \ 3^- \end{array}$$

At the first step, we make a comparison between the first gene of P_1 and that of P_2 , so we compare 1^+ with 2^+ according to the global precedence vector. In the global precedence vector, the location 1^+ takes precedence over location 2^+ , so 1^+ is the first gene of the offspring that inherits the gene from P_1 . With regard to P_2 , the first gene 2^+ exchanges the position with the gene 1^+ , so that we can maintain the validity of the route.

The gene with the earlier precedence is placed into the offspring C and genes are swapped to maintain validity if necessary. Continuing the process until C is filled with genes. Finally, a route is produced.

$$1^+ \ 3^+ \ 2^+ \ 2^- \ 1^- \ 4^+ \ 3^- \ 4^-$$

Effectively, MX1 produces a child that is close to the order of the global precedence.

With the same global precedence vector as the MX1 operator, the MX2 operator on two chromosomes also produces a single offspring. The contents of P_1 and P_2 are also the same as the preceding example.

The process is similar to merge two sorted vectors. The first gene of P_1 is found to be prior to the one of P_2 according to the global precedence vector. The prior gene 1^+ is placed in the offspring C . Then, the first gene of P_1 is removed in both individuals, hence the 1^+ is removed from both P_1 and P_2 . The second gene of P_1 is compared with the first gene of P_2 according to the global precedence vector. The first gene of P_2 (2^+) is found to be the prior gene. Again the prior gene is placed into the offspring C , and removed from the both chromosomes. By continuing the process in this fashion, we can get an offspring C shown below.

$$1^+ \ 2^+ \ 3^+ \ 1^- \ 2^- \ 4^+ \ 3^- \ 4^-$$

This operator will move the gene with the lowest precedence near the end of the chromosome.

3.3.3 Mutation

The probability of mutation rate is very small, and in our approach, it is not a fixed value. Mutations will be applied when the offspring is the same as one of its parents. 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 three mutation operators that can be applied to the single-vehicle PDPTW problem.

- (1) Two genes are selected randomly, and their positions are interchanged.
- (2) Randomly two cut sites are chosen, and the order of the sub-route specified by the genes is inverted.
- (3) If the vehicle arrives at the i th stop and violates the constraints, we will disturb the order of the genes within the first i th sub-route.

The first operator will create a new route which has four different edges from its original route. The second mutation operator is similar as the 2-opt move in TSP; the difference between the new and the original route is two edges. Finally, the third mutation operator rearranges a specific sub-route; such a change may affect the solution cost significantly.

4 Experimental Results

We have tested the hybrid genetic algorithm on five problems consisting of 10, 20, 30, 40 and 50 tasks, respectively. In these problems, the pickup locations and the delivery locations are randomly placed in a rectangular grid. The locations of each task are associated with a time window, which is generated randomly. The Euclidean distance is used to measure the distance between a pair of locations.

In our experiments, the global precedence vector of MX1 and MX2 is based on the lower bound of time windows. We test the four crossover operators associated with three different crossover rates, the rates are 0.45, 0.6 and 0.75. The population size of each problem is the number of task multiplied by 5. That is, if the number of task is 40, the total number of location is 81 and the population size is 200. The value of each factor in the objective function will be $\alpha_1 = 10$, $\alpha_2 = 50$, $\omega_1 = 1$, and $\omega_2 = 2$.

At first, we would like to make a comparison among the crossover operators without the mutation operators, in order to study the pure relative performance of the crossover operators. In Figure 2, the mutation operators are not applied to the single-vehicle PDPTW problem. The solution cost is based on $Z_{objective}$ where $\omega_1 = \omega_2 = 1$. In Figure 2, for each generation, the values of solution cost are the average in 20 runs.

Figure 2 shows the MX1 could achieve better solution cost than the other crossover operators do. According to the Figure 2, OX is not a proper operator for the single-vehicle PDPTW problem. OX creates offspring which inherits sub-tours of its parents. Such a new route is analogous to its parent routes. The solutions were trapped to similar routes, and therefore

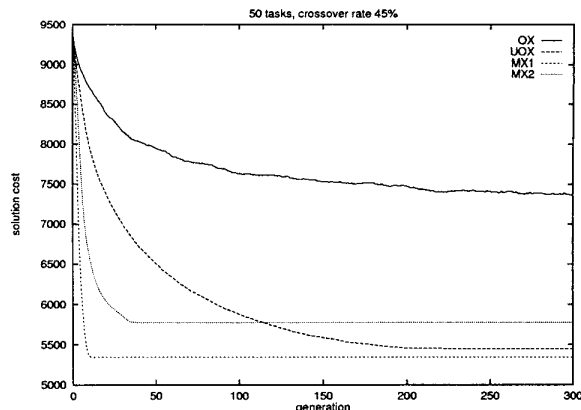


Figure 2: The comparison of crossover operators

feasible solutions may not be found. As to MX2, a location with the lowest precedence is moved to near the end of the routes, and the solution may be stuck to a local minimum. The solution cost of UOX is approximate to the solution cost of MX1 when the generation is larger than 200, but MX1 achieves the same solution cost before the 50th generation. For the real-time requests, the UOX may not be suitable for the dynamic single-vehicle PDPTW problem.

We apply the three mutation operators with the MX1 crossover operator. Table 1 describes the ratios of best costs to the optimal cost of the 10, 20, 30 and 40 task problems. A dynamic programming[4] accomplished the optimal solution of the single-vehicle PDPTW problems. The optimal solutions of 10, 20,

Table 1: Relative optimality of the solutions

task no.	cr. rate	mutation operators				optimal value
		-	(1)	(2)	(3)	
10	0.45	1.000	1.000	1.000	1.000	1135
	0.60	1.000	1.000	1.000	1.000	
	0.75	1.000	1.000	1.000	1.000	
20	0.45	1.000	1.000	1.000	1.000	1794
	0.60	1.000	1.000	1.000	1.000	
	0.75	1.000	1.000	1.000	1.000	
30	0.45	1.033	1.000	1.000	1.033	2602
	0.60	1.022	1.000	1.000	1.033	
	0.75	1.033	1.000	1.000	1.033	
40	0.45	1.053	1.000	1.000	1.051	4082
	0.60	1.051	1.000	1.000	1.051	
	0.75	1.040	1.000	1.000	1.051	
50	0.45	5336	5247	5247	5336	-
	0.60	5341	5249	5257	5336	
	0.75	5341	5266	5276	5336	

30 and 40 tasks are listed in the last column of Table 1. The mutation operator '-' means that no mutation operator is applied to the genetic algorithms. As to the problem of 50 tasks, the results are shown in the last three rows of Table 1. Most of exact algorithms applied to vehicle routing problems can not solve relatively larger problems, and we can not obtain the optimal values neither. From the results of Table 1, it shows the mutation1 and mutation2 attain better solution cost than the other mutation operator does.

Figure 3 shows the average solution costs of the hybrid genetic algorithms are better than the average costs of the traditional genetic algorithms. A dynamic

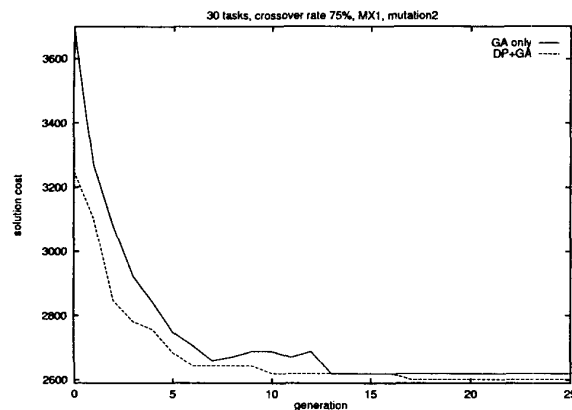


Figure 3: DP+GA vs. GA only

programming is applied to generate the initial population instead of creating it randomly. Since the initial population filtered out some of the infeasible routes and put the initial population contents more feasible sub-routes, the hybrid genetic algorithm always obtain the lower solution cost than the traditional genetic algorithms.

5 Conclusion

In this paper, we show that the hybrid genetic algorithm solves the dynamic single-vehicle PDPTW problem. With such approach, that takes advantage of both dynamic programming and genetic algorithms. The approach enables dynamic programming to achieve real-time performance and genetic algorithms to approximate optimal solutions. The initial population created by the dynamic programming instead of generating it randomly. The dynamic programming passes the unfinished routes to genetic algorithms to accomplish the real-time performance. A good initial population of the genetic algorithms is improved by the dynamic programming.

The hybrid approach can generate incrementally better solutions at any time, which is essential for dealing with dynamic task requests. Comprehensive experiments using an efficient and flexible implementation of the hybrid genetic algorithms were performed. The experimental results showed that the hybrid approach could produce near-optimal solutions for problems of sizes up to 25 percent bigger than what can be solved previously by dynamic programming. In addition, the hybrid approach may find sub-optimal solutions for dynamic vehicle routing problems of any size.

References

- [1] 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.
- [2] 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, Amsterdam, 1988.
- [3] 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*, 6(3 & 4):301-325, 1986.
- [4] H. N. Psaraftis. A dynamic programming solution to the single vehicle many-to-many immediate request dial-a-ride problem. *Transportation Science*, 14(2):130-154, 1980.
- [5] H. N. Psaraftis. An exact algorithm for the single vehicle many-to-many dial-a-ride problem with time windows. *Transportation Science*, 17(3):351-357, 1983.