Effective Heuristics to Solve Pickup and Delivery Problems with Time Windows

Vincent Tam and Lois C.Y. Tseng
Dept of EEE, University of Hong Kong, Pokfulam, Hong Kong.
vtam@eee.hku.hk

Abstract

Pickup and delivery problem with time windows (PDP-TW) is a challenging scheduling problem for which each delivery is coupled with a pickup request. Metaheuristic search techniques like the tabu search have been used to solve PDP-TW. In this paper, we investigated a min-conflicts based micro-genetic algorithm combining some interesting construction heuristic, namely the Align-Fold or Boomerang, and repair heuristics including a new Swap operator and a modified billiard operator to effectively solve PDD-TW. Our results compared favorably against those of a tabu-embedded metaheuristic search on a set of modified Solomon's test cases. More importantly, our proposed heuristics can easily be integrated into many search schemes for solving other complex scheduling problems.

1. Introduction

Delivery(-only) problem with time windows (DP-TW) notably represents a class of challenging delivery problems with a wealth of published results [1, 9, 10] in both areas of Artificial Intelligence [3, 8] and Operations Research [2, 9]. The main task is to effectively schedule a fleet of delivery vehicles in order to satisfy a number of customers' requests with user-specified service time windows, thus constraining each delivery to occur within a limited period. Extending from DP-TWs, the pickup and delivery problems with time windows (PDP-TW) [6, 7, 11], with additional time constraints for pickups to be coupled with the customers' delivery requests, represent a more general and challenging class of delivery problems with wider applicability to modern logistics applications for the land, sea or air transport. Examples of PDP-TWs include the dial-a-ride application and bus scheduling.

Heuristics have been widely used for vehicle routing due to their effectiveness. The push forward insertion heuristic (PFIH) [10] was a route construction heuristic proposed by Solomon to handle DP-TW. Basically, PFIH compares the cost of inserting a new customer into the current route with the lowest possible cost against that of creating a new route. A modified PFIH was proposed in [7] to handle PDP-TW.

For vehicle routing, many local search methods like the tabu search [2] or genetic algorithms [3, 4] often use the PFIH or its variants to construct an initial solution before applying any heuristic operators to optimize the objective value of the current solution. Tabu search (TS) [2] is an example search strategy, with the use of short-term memory to avoid cycling, to solve many practical combinatorial problems including timetabling [5], integrated circuit design and vehicle routing [2]. Besides, genetic algorithm (GA) [3, 4] is a type of adaptive heuristic search based on natural evolution. A population of chromosomes is generated and continuously modified by some genetic operators to produce offsprings for another new generation until a predefined stopping criterion is reached, or a locally optimal solution is found. An example of GA is the GIDEON [4] algorithm.

In this paper, we adapted a min-conflicts based micro-genetic algorithm [4, 8] to solve PDD-TW. Moreover, we proposed two interesting construction heuristics namely the Align-Fold and Boomerang as alternatives to the adapted PFIH, and a new Swap operator and a modified billiard operator to effectively solve the PDP-TW. In particular, the billiard operator, resembling the billiard ball movement originally for repairing a timetable [5], was adapted so that an assigned pair of pickup-and-delivery requests could be knocked off by an unassigned lecture only when the already assigned pair can find appropriate empty slots (i.e. holes) in any existing route for insertion. Our proposed initialization and repair heuristic operators were integrated into 6 different search algorithms. Their results compared favorably against those of a tabu-embedded metaheuristic search [7, 9] on a set of modified Solomon's test cases. More importantly, our proposed initialization and repair heuristics were so generic and thus easily integrated into many search schemes to possibly solve other optimization problems.

The paper is organized as follows. Section 2 reviews the preliminary concepts and formal definitions about the PDP-TW to facilitate our subsequent discussion. Section 3 describes newly designed initialization heuristics such as Align-Fold and Boomerang, and repair operators including the modified billiard move and Swap. Section 4 gives an empirical evaluation on the performance of our proposals against those of Li & Lim's metaheuristic approach [7]. Lastly, we conclude our work in Section 5.

Proceedings of the 15th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'03)
1082-3409/03 $17.00 © 2003 **IEEE**

2. Pickup and Delivery Problems with Time Windows (PDP-TW)

Similar to DP-TW, pickup and delivery problems with time windows (PDP-TW) are constrained optimization problems [9]. The formal definition of PDP-TW [6, 7] is stated as follow. Given a node set $N = \{n_0, n_1, n_2, n_3, \ldots, n_m\}$ where n_0 always denotes the depot, n_1 to n_m denote delivery or pickup locations for customers' requests, and the last index m is always an even number, each individual customer request is represented by a pair of delivery and pickup locations. Each delivery or pickup location n_i where $i \neq 0$ is associated with a customer demand q_i such that $q_i > 0$ for a pickup location whereas $q_i < 0$ for a delivery location, a service time s_i, that is the duration required to effectively service the customer demand at that location, and an associated service time window $[e_i, l_i]$ where e_i and l_i denote the earliest and latest time to start the service. The delivery and pickup demand q_i and q_j belonging to the same customer will have the same magnitude so that $q_i + q_j = 0$ for ease of analysis. Besides, for any possible edge $<n_i, n_j>$, both the non-negative distance d_{ij} and required travel time t_{ij} are specified. However, it should be noted that due to the time-window constraints, **not** every possible edge is a feasible edge to construct a feasible route for any vehicle when solving the DP-TW or PDP-TW. In other words, only those edges $<n_i, n_j>$ that satisfy their corresponding time-window constraints as $t_{oi} + s_i + t_{ij} \leq l_j$, restricting the vehicle concerned to arrive at or before the latest service time l_j after traveling from the depot to n_i to n_j with its completion of service at n_i, should be considered.

In addition to the time-window constraints, several problem constraints must not be violated. First, each vehicle has a limited capacity C that cannot be exceeded. Each vehicle must carry an amount less than or equal to C. Second, all vehicles depart from and return to the same depot n_0, and share the same constraints time window $[E, L]$, where E denotes the time a vehicle must have left the depot, and L denotes the time a vehicle must have returned to the depot. Third, a customer can only be serviced within the associated service time window $[e_i, l_i]$. That is, if a vehicle reached the customer earlier than e_i , the vehicle has to wait until e_i. Lastly, the coupling constraints request that every pair of pickup and delivery locations must be serviced by the same vehicle while the precedence constraints specify that the pickup location must be serviced first. Clearly, the objective functions vary depending on different applications. For instance, in the dial-a-ride application, a common objective is to minimize the inconvenience (often measured in term of the total waiting time) as caused by the service to occur earlier or later than the expected time. Following [7], we consider in this paper an objective cost function with 4 parameters in descending order of importance as follows: the number of vehicles used, the total traveling cost, the total schedule duration, and the drivers' total waiting time.

3. Useful Heuristics to Solve PDP-TWs

Generally speaking, heuristics play a very significant role in affecting the overall performance of a local search method. When handling the DP-TW or PDP-TW, heuristic operators can be classified as intra-route or inter-route operators [4]. Examples of intra-route operators are *two-opt* operator and *rearrange* operator, while examples of inter-route operators are *exchange* operator and *shift* operator. To facilitate our subsequent discussion, we define an operation as "valid" if and only if the solution produced violates no constraint and the cost of the solution is reduced at the same time. Basically, all the above intra- or inter-route operators aim to produce valid operations. However, due to the 'coupling' nature of delivery and pickup nodes in PDP-TW, these heuristics operators need to be implemented in a slightly different logical flow [7] to effectively tackle the PDP-TW. Basically, the *two-opt* operator works by exchanging positions of two customers in the route whereas the *rearrange* operator re-arranges a pickup-delivery customer pair into more cost effective positions within the same route. *Shift* operator removes a pickup-delivery customer pair from one route, and then inserts the customer pair into another route. For *exchange* operator, it involves 2 routes, route A and route B. From each route, a pickup-delivery customer pair is removed. The customer pair originally in route A is now inserted into route B, while the customer pair originally in route B is now inserted into route A.

Besides these commonly used heuristics, we consider two interesting initialization and repair heuristics for tackling PDP-TWs in the subsequent subsections.

3.1. Initialization Heuristics

When handling PDP-TWs, by firstly sorting customer-pairs based on their combined distance from the depot, we propose the Align-Fold and Boomerang route construction heuristics as follows.

- **The Align-Fold Initialization Method**

The initialization heuristics involves 2 phases: Align and Fold. In the 1st Align stage, customer pairs are inserted into a vehicle one by one in descending order of combined distance from the depot. When a customer pair cannot be inserted into the current route due to constraints violation, a new route will be created. The insertion process continues until all customer pairs are routed.

After the Align Stage, there may be gaps of different sizes in the beginning few routes, that can possibly be filled up by customer-pairs (of smaller distance from the depot) shifted from the last few routes. Such shifting and filling operation is called folding. Folding requires a boundary line to be set among the vehicles. Following the strategy to preset the number of "virtual" vehicles for constructing the initial solution in [9], we refer to research results produced by Li and Lim [7] to determine the boundary line for each case. For each customer-pair to be folded, we compute the folding cost for every possible insertion position in any upper vehicle *i* as *i*(Δ scheduled duration per route + Δ distance traveled by the vehicle per route)*. The position with the least folding cost would be chosen. Obviously, we will iteratively try for more folding at a higher boundary line whenever possible.

o **The Boomerang Initialization Method**

In Boomerang, sorted customer-pairs of the combined distance from depot are divided into classes. Each class would hold *N* pairs of customers, where *N* refers to best number of vehicles obtained previously [7]. Class A stores the first *N* customer pairs with longest combined distance, class B store the second longest batch, etc. Under the Boomerang approach, customer-pairs from Class A are always inserted into the first positions of all *N* vehicles. The first customer-pair in class A will be inserted into Vehicle *1* until the last pair inserted into Vehicle *N*. After that, each vehicle would take turn to pick up a customer pair from class B. Hopefully, all the constructed routes will exhibit boomerang-like paths as illustrated in the coordinate plot below:

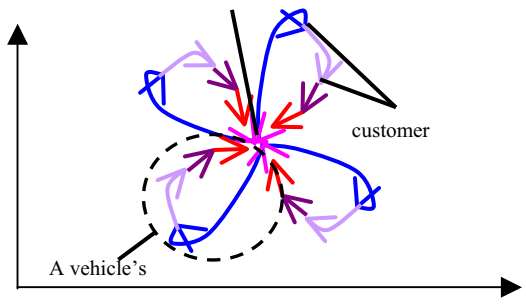

Figure 1. Resulting boomerang-like routes

Basically, the Boomerang approach tries to minimize the distance traveled by each vehicle by servicing some possibly nearby customer-pairs on its returning path. It should be noted that each single point (location) in Figure 1 actually corresponds to a pair of customer requests. Besides, any customer pairs that cannot be fitted into the pre-assigned vehicle will be inserted in any possible positions in any other existing vehicle; otherwise, a new vehicle would be created.

3.2. Repair Heuristics

In the following, we consider two interesting repair heuristics, namely the modified Billiard and Swap operators, to iteratively improve the objective value of the current routing plan until a local minimum is reached.

o **The Modified Billiard Operator**

In [5], an interesting billiard operator resembling the movement of billiard balls was applied to repair timetables. To handle PDP-TW [6], we propose to modify the original billiard operator as inserting a customer-pair #1 originally in route A into positions already assigned to another customer-pair #2 in route B if and only if a pair of vacant space in route C can accommodate the customer pair #2 so as to yield a solution with lower objective cost after this move. Our modified billiard operator, not restricted to the original positions of customer-pair #2, will also consider other feasible position in route B if the resulting move is still a "valid" operation as previously defined.

o **The Swap Operator**

Swap operation works by a substantial modification of the current solution. From each vehicle in the fleet, the swap operator will randomly pick up a few pairs of customers, remove them from the vehicle and add them into a relocation pool. The number of customer pairs to be removed from each vehicle depends on fleet size. The major consideration is to keep the relocation pool having roughly the same number of customers in each swap. As an example, in our prototype implementation, we arbitrarily set the swap operator to remove around 1/5 of total number of customers into the relocation pool. When there is any empty vehicle after removal of customers, the vehicle would be removed from the fleet. Then, the operator will randomly choose a pair of customers from relocation pool and insert them into any vehicle based on the objective cost function described in Section 2. In case there is no possible position in any route to insert the customer pair(s) in the relocation pool, a new vehicle would be created.

4. An Empirical Evaluation

To demonstrate their feasibility, our proposed initialization and repair heuristics were integrated into six different search strategies including a micro-genetic algorithm [4], that is a GA with small population size, using the adapted *exchange* operator *(MGA(Ex)),* another micro-genetic algorithm using our modified billiard operator *(MGA(Bil)),* a simple search scheme *AFS* using the Align-Fold initialization method and the Swap operator iteratively until no further improvement,

AFS+RandOp as a variant of *AFS* randomly applying the *two-opt, rearrange, shift* or *exchange* operator as post-optimization operations, another local search scheme BS using the Boomerang initialization method and the Swap operator until no possible improvement, and lastly *BS+RandOp* as a variant of *BS* employing similar post-optimization strategy. All algorithms were implemented using the Java Development Kit (JDK) Version 1.4. And all tests were run on a desktop computer with Intel Pentium IV processor of *2 GHz*, *512 MB* RAM, and a hard disk of *20 GB* space. The operating system used was the Microsoft Windows 2000.

The dataset used is *56* modified problem instances [7] of the well-known Solomon's test cases [10]. Each problem instance has around 100 customers. There are totally 6 distinct classes, namely *LC1, LC2, LR1, LR2, LRC1*, and *LRC2*. *'LC'* refers to cases with clustered distribution of customers, *'LR'* refers to cases with uniform distribution of customers, and *'RC'* refers to mixed customers types. '1' refers to small vehicle capacity whereas '2' refers to large vehicle capacity.

Table 1 summarizes the overall performance of our different proposals as compared to Li & Lim's published results [7] over the 56 modified cases.

Our proposal compared to Li & Lim's work	Win	Tie	Lose
MGA(Ex)	*0*	*16*	*40*
MGA(Bil)	*2*	*20*	*34*
AFS	*5*	*30*	*21*
AFS+RandOp	*4*	*30*	*22*
BS	*7*	*29*	*20*
BS+RandOp	*0*	*15*	*41*

Table 1. Summary of each search approach as compared to Li & Lim's published results

In terms of number of wins, ties and loses, BS was found to be the best with 7 obtained results better than those of Li & Lim's published results over the 56 test cases. *AFS* and *AFS+RandOp* were the second and third best search algorithms among our 6 proposals. The number of breakthroughs and ties of *AFS+RandOp* were less than that of *AFS*. This is possibly due to the randomized nature of our proposed Swap operator. The percentage of breakthroughs reported in the results of *BS* was relatively more than those of *AFS* base approaches. This could be explained by a better initial solution generated by the Boomerang method than that of the Align-Fold initialization method. Table 2 shows the detailed improvement of *BS* over Li & Lim's metaheuristic approach on the 7 specific test cases of PDP-TW. For a more detailed analysis on these winning cases, refer to [11].

Boomerang and Swap	**Fleet size**	**Distance**	**Scheduled Duration**	**Waiting Time**
LC 204	3	590.60 (↓0.10%)	9590.60 (↓0.01%)	0.00
LR 109	11	1208.96 (↓2.50%)	2282.20 (↓1.26%)	73.23
LR 201	4	1253.23 (↓0.84%)	3495.81	1242.57
LR 209	3	930.59 (↓0.69%)	2415.99 (↓0.68%)	485.41
LRC 201	4	1456.37 (↓0.86%)	3358.41	902.04
LRC 206	3	1159.03 (↓0.33%)	2444.87	285.83
LRC 207	3	1064.40 (↓25.28%)	2419.86 (↓1.69%)	355.46

Table 2: Detailed improvement of BS over Li & Lim's published results on the 7 test cases

Lastly, since the shorter the total execution time required, the more acceptable the algorithm should be. Table 3 gives a summary of the total execution time (in CPU seconds) for each of our proposed search approach. In terms of medians, *BS, AFS, AFS+RandOp* and *BS+RandOp* can manage to complete execution in around *1.5* CPU minutes. However for *MGA(Ex)*, the median execution time was more than 3.3 times to that of the quickest *BS* algorithm. For *MGA(Bil)*, it was even more than 10 times of that of the *BS* algorithm. As a whole, this will definitely make the MGA based algorithms less favorable as compared to our other proposals. In essence, these results also induce the need to modify the two MGA approaches for better efficiency in our future work.

	Median Exec. Time	STD.DEV.
MGA (Ex)	217	8816.35
MGA (Bil)	556.5	22942.81
AFS	51.5	47.89
AFS+RandOp	62	1144.60
BS	49.5	50.23
BS+RandOp	90.5	9240.49

Table 3. Median and STD.DEV. of the performance of our search approach

5. Concluding Remarks

In this paper, we considered a formal definition [7] of the more general and challenging class of delivery problems, namely the pickup and delivery problems with time windows (PDP-TW). Moreover, we reviewed 4 commonly used intra- and inter-route heuristic operators, namely the two-opt, rearrange, exchange and shift operators [4], adapted to handle the PDP-TW effectively. These heuristic operators have been integrated into many local search methods including a tabu-embedded metaheuristic search method proposed by Li & Lim [7].

Independently, we proposed two interesting construction heuristics namely the Align-Fold and Boomerang as alternatives to the commonly adopted push forward insertion heuristic (PFIH) [10]. In addition, we considered a new Swap operator and a modified billiard operator for integration into our proposed search algorithms. In general, our obtained results compared favorably against those of Li & Lim's tabu-embedded metaheuristic search proposal on a set of modified Solomon's test cases. Besides, no matter what particular aspect we focused on like the number of winning cases, the values of cost parameters and the total execution time, our 3 proposed search methods *AFS+RandOp, AFS* and *BS* always bettered than our other proposals with *BS* consistently giving the best overall performance. Furthermore, *BS* also outperformed Li & Lim's metaheuristic search methods on 7 test cases. This demonstrates the amazingly improving ability of the Swap operator over the other heuristic operators we considered in this paper.

In fact, there can be many possible directions for future investigations. As PFIH shows its power in generating initial solutions of relatively good solution quality when compared to our other proposals, we can consider to combine the modified PFIH with our Swap operator to look for better performance. Besides, operators like the exchange and modified billiard move are expensive in computational costs, implying that we may try to redesign the heuristic operators more careful for better search efficiency. Lastly, to improve the Align-&-Fold or Boomerang route construction heuristics, rather than simply grouping all customers according to combined distance from depot, we may try to group customers by taking into account of the time windows as well. This is to divide a day's schedule into *N* categories and then consider the combined distance from depot while ensuring no two customer-pairs can be overlapping in their service time windows during the assignment to the different vehicles.

References

[1] Olli Braysy, *"Genetic Algorithms for the Vehicle Routing Problem with Time Windows"* Special issue on Bioinformatics and Genetic Algorithms , Arpakannus 1/2001.

[2] Fred Glover, *"Tabu Search-Part I"*, ORSA Journal on Computing Vol.1, No. 3. Summer 1989.

[3] John H. Holland, "*Adaptation in Naturaland Artificial Systems*.", University of Michigan Press, Ann Arbor, 1975.

[4] Jiachang Jee, *"Solving Vehicle Routing Problems with Time Windows using Micro-Genetic Algorithms"*, UROP report (supervised by Dr. Vincent Tam), School of Computing, The National University of Singapore, 1999/2000.

[5] Kazuya Kancko, Masazumi Yoshikawa, and Yoichiro Nakakuki, *"Improving a Heuristic Repair Method for Large-Scale School Timetabling Problems"*, Principles and Practice of Constraint Programming - CP'99, 5th International Conference, Alexandria, Virginia, USA, October 11-14, 1999.

[6] H.C. Lau and Z. Liang, *"Pickup and Delivery with Time Windows: Algorithms and Test Case Generation",* 13th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'01) November 07 - 09, 2001.

[7] Haibing Li, Andrew Lim, "A Metaheuristic for the Pickup and Delivery Problem with Time Windows" 13th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'01) November 07 - 09, 2001.

[8] S. Minton, M. Johnston, A. Philips, and P. Laird. *"Minimizing conflicts: a heuristic repair method for constraint satisfaction and scheduling problems."* Artificial Intelligence, 1992.

[9] P.Prosser and P. Shaw, *"Study of greedy search with multiple improvement heuristics for vehicle routing problems."* Technical Report, RR/96/201, Department of Computer Science, University of Strathclyde, Glasgow, Jan 1997.

[10] M.M. Solomon, *"Algorithms for the vehicle routing and scheduling problem with time windows"* Oper.Res. 35:254-265, 1987.

[11] Lois C.Y. Tseng, *"Vehicle Routing Problems with Pickup and Delivery using Time Windows"*, Final-Year Project Report, Dept. of EEE, HKU, 2002/03.