Omega 61 (2016) 155–166

Contents lists available at ScienceDirect

Omega

journal homepage: www.elsevier.com/locate/omega

Pareto mimic algorithm: An approach to the team orienteering problem ☆

Liangjun Ke [a,*], Laipeng Zhai [a], Jing Li [b], Felix T.S. Chan [c]

[a] The State Key Laboratory for Manufacturing Systems Engineering, Xian Jiaotong University, Xi'an 710049, China
[b] The State Key Laboratory of Astronautic Dynamics Xi'an Satellite Control Center, Xi'an 710043, China
[c] Department of Industrial and Systems Engineering, The Hong Kong Polytechnic University, Hung Hom, Hong Kong

ARTICLE INFO

Article history:
Received 20 August 2013
Accepted 4 August 2015
Available online 12 August 2015

Keywords:
Vehicle scheduling
Vehicle routing problem
Team orienteering problem
Pareto dominance

ABSTRACT

The team orienteering problem is an important variant of the vehicle routing problem. In this paper, a new algorithm, called Pareto mimic algorithm, is proposed to deal with it. This algorithm maintains a population of incumbent solutions which are updated using Pareto dominance. It uses a new operator, called mimic operator, to generate a new solution by imitating an incumbent solution. Furthermore, to improve the quality of a solution, it employs an operator, called swallow operator which attempts to swallow (or insert) an infeasible node and then repair the resulting infeasible solution. A comparative study supports the effectiveness of the proposed algorithm, especially, our algorithm can quickly find new better results for several large-scale instances. We also demonstrate that Pareto mimic algorithm can be generalized to solve other routing problems, e.g., the capacitated vehicle routing problem.

1. Introduction

Vehicle routing problem (VRP) is a core problem in logistics. In recent years, a widely studied variant of the VRP is the team orienteering problem (TOP). In this problem, a fleet of vehicles are scheduled to serve a set of nodes (or customers), each of which is associated with a reward. Once a node is served by a vehicle, the corresponding reward will be received. The objective is to maximize the total received reward while the travel time of each route must be not more than a time limit. The TOP was first formulated for team orienteering, which is a kind of outdoor sport. It can be used to model a broad range of real-life applications such as tourist trip planning [1] and sportsman recruitment [2]. An interesting application is unmanned aerial vehicle (UAV) mission planning [3] in which some UAVs attached with various sensors are required to patrol a set of targets. Each target is assigned a profit, and the planner aims to maximize the total received profit.

The TOP is a multi-level combinatorial optimization problem [2]. The first level is to choose a subset of nodes to visit, the second level is to assign the chosen nodes to each vehicle, and the third level is to find a route through the nodes assigned to each vehicle. It is well known that the TOP is NP-hard [2]. So far several exact algorithms [4–7] have been developed. However, unless P=NP, there is no algorithm which can find an optimal solution within time polynomial in the size of customers. An alternative approach to the TOP is metaheuristic, which aims to yield a satisfactory solution within reasonable time. To date, ant colony optimization (ACO) [8], guided local search [9], large neighborhood search [10], memetic algorithm (MA) [11], particle swarm optimization [12], path relinking [13], tabu search [14,15], and variable neighborhood search [15] have been applied to the TOP. The interested reader is referred to [1] for a survey. Among the current metaheuristics, the particle swarm optimization algorithm in [12] ranked first when testing on the instances of [2]. Moreover, the results on larger new generated instances demonstrate that their algorithm is very effective though it consumes relatively long running time.

In the design of a metaheuristic [16], solution generation is critical to the performance. When generating new solutions, a metaheuristic employs specific operators by using one or more incumbent solutions. These operators can be categorized as constructive or non-constructive. For example, ACO [8] adopts a construction procedure which depends on pheromone trails and heuristic information. The limitation is that it is somewhat complex to maintain the pheromone trails. MA [11] uses a non-constructive operator, i.e., crossover to generate new individuals, each of which is represented by an ordered sequence of the customers. It decodes a sequence into a solution by a so-called split [17] or route first-cluster second approach [18]. To make a metaheuristic work better, it is challenging to investigate simple operators so as to exploit the incumbent solutions better.

☆This manuscript was processed by Associate Editor Yagiura.

* Corresponding author.
E-mail address: kelj163@163.com (L. Ke).

http://dx.doi.org/10.1016/j.omega.2015.08.003
0305-0483/

Solution update rule determines the criterion on choosing new incumbent solutions. As far as we know, the update rules of the current metaheuristics for the TOP are quality dependent. That is, new incumbent solutions are chosen based on their objective values. It is no doubt that one will expect that those chosen solutions are of high-quality and low-similarity, thereby leading to explore potential and wide search areas. It is therefore beneficial to highlight the similarity in the update rule. In the literature, multi-criteria decision has proven to be useful to preserve the diversity of the incumbent solutions [19,20]. Here we adopt this technique to deal with the TOP for the first time.

In this paper, a metaheuristic, called Pareto mimic algorithm, is proposed. The main contributions are summarized as follows: (1) It uses a new operator, called mimic operator, to generate a new solution by imitating an incumbent solution. Via a parameter, this operator easily controls the similarity between the generated solution and the incumbent solution. (2) It updates a population of incumbent solutions based on Pareto dominance. In fact, to determine, incumbent solutions can be modeled as a multi-criteria decision problem [21]. This prompts us to define multiple indicators to measure each candidate solution. Since a solution may be not better than others in terms of all indicators, Pareto dominance [21], a concept that was first used in economics, is adopted for updating incumbent solutions. Due to the mimic operator and Pareto dominance based update rule, the proposed algorithm is named after *Pareto mimic algorithm* (PMA). (3) It adopts a new operator, called swallow operator, to improve a solution. This operator attempts to swallow (or insert) an infeasible node and then repair the resulting infeasible solution. In contrast to local search, the swallow operator allows the search to tunnel through the infeasible solution area. The idea of tunneling through the infeasible solution area has also been used in [13,15].

The remainder of this paper is structured as follows. Section 2 presents a description of the TOP. Section 3 presents the proposed algorithm. Section 4 presents the experimental study. Section 5 discusses how to generalize PMA for other routing problems. Finally, Section 6 concludes the main results.

2. A description of the top

The TOP is defined on a complete graph $G=(V,E)$, where $V=\{0,1,\cdots,n+1\}$ is the set of nodes and $E=\{(i,j)|i,j\in V\}$ is the set of edges. Let c_{ij} be the travel time of edge $(i,j)\in E$ and r_i be the reward of node i. A feasible path must start at node 0 and finish at node $n+1$, and its travel time cannot exceed a time limit $T_{\max}$. Let Ω be the set of all feasible paths. There are m vehicles.

Let $R(x_k)$ be the total received reward of a path $x_k\in\Omega$. Symbol $a_{ik}=1$ if path $x_k\in\Omega$ visits customer i ($1\le i\le n$), otherwise $a_{ik}=0$. Variable $y_k=1$ if path $x_k\in\Omega$ is traveled, otherwise $y_k=0$. The TOP can be stated as follows [5]:

$$\begin{aligned}
&\text{maximize} && \sum_{x_k\in\Omega} R(x_k)y_k \\
&\text{subject to} && \sum_{x_k\in\Omega} a_{ik}y_k\le 1, \quad 1\le i\le n \\
& && \sum_{x_k\in\Omega} y_k\le m \\
& && y_k\in\{0,1\}, \quad x_k\in\Omega
\end{aligned} \tag{1}$$

In this formulation, the goal is to maximize the total received reward. The first constraint means that each node only can be visited at most once, and the second constraint ensures that there are at most m paths in a feasible solution.

3. The proposed algorithm

Pareto mimic algorithm, denoted as PMA, is an iterative solution technique. At first, N incumbent solutions are initialized. The best-so-far solution x_b is set as the one with the largest objective value among these N solutions where the objective value of a solution x, denoted by $F(x)$, is the total received reward of those paths of x. At each step, it works as follows: starting from each incumbent solution, a new solution is generated by the mimic operator. Subsequently, local search and the swallow operator are employed to improve it. By using the new generated solutions and those old incumbent solutions, the set of incumbent solutions, denoted as IS, is updated according to Pareto dominance. The best-so-far solution x_b is also renewed. PMA terminates when a stopping condition is satisfied. The main procedure of PMA is given in Algorithm 1.

Algorithm 1. The main procedure of the proposed algorithm.

Input: A TOP to be solved
Output:the best-so-far solution x_b
1: set $IS:=\varnothing$
2: set x_b as the solution which contains no node
3: **for** $i=1$ **to** N **do**
4: $x:=$ **Initialization**() /*see Section 3.2*/
5: set $IS:=\{x\}\bigcup IS$
6: replace x_b by x if $F(x)>F(x_b)$
7: **end for**
8: **while** the termination condition is not satisfied **do**
9: set $U:=$ the size of IS
10: set $Q:=\varnothing$/*Q is an auxiliary set*/
11: **for** $i=1$ **to** U **do**
12: $x:=$ **MimicOperator**(the ith solution of IS)/*see Section 3.3*/
13: $x:=$ **LocalSearch**(x) /*see Section 3.4*/
14: $x:=$**SwallowOperator**(x) /*see Section 3.5*/
15: set $Q:=\{x\}\bigcup Q$
16: replace x_b by x if $F(x)>F(x_b)$
17: **end for**
18: $IS:=$ **Update** ($Q\cup IS$)/*see Section 3.6*/
19: **end while**

3.1. Notations

(1) *Static preference value*: This value indicates the favorite degree of transiting from one node to another node. Suppose the current node is i, the static preference value of node j is defined as r_j/c_{ij}.

(2) *Dynamic preference value:* Given a solution x, let $\Delta T(x,i,k)$ be the increment of travel time caused by inserting i into the kth best position. The dynamic preference value of node i, denoted as $D(x,i)$, is defined as follows:

$$D(x,i)=(\Delta T(x,i,3)-\Delta T(x,i,1))\cdot r_i^u \tag{2}$$

where $u\in(0,1]$ is a uniformly distributed random number. $\Delta T(x,i,3)-\Delta T(x,i,1)$ is the difference of travel time obtained by inserting i into the third best and the best position. We also tested other possibilities, e.g., the second best position instead of the third best one in Eq. (2), but the presented choice gives us the best performance (see Section 4.4). In some references, this difference is also called regret value [22,23]. We adopt this definition of regret value since it is effective and cheap in computation. If there is only one position to be inserted, $\Delta T(x,i,3)$ is set as a sufficiently large value, which means that it is urgent to insert node i. A larger dynamic preference value indicates that the node is more favorable. In contrast to the static preference value, the dynamic preference value depends on the starting solution and it is randomized.