Computers & Operations Research 40 (2013) 1306–1313

Contents lists available at SciVerse ScienceDirect

Computers & Operations Research

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

Reoptimizing the rural postman problem

C. Archetti, G. Guastaroba *, M.G. Speranza

Department of Quantitative Methods, University of Brescia, Italy

ARTICLE INFO

Available online 20 December 2012

Keywords:
Reoptimization
Rural postman problem
Heuristic algorithms
Worst-case analysis

ABSTRACT

Given an instance of the Rural Postman Problem (RPP) together with its optimal solution, we study the problem of finding a good feasible solution after a perturbation of the instance has occurred. We refer to this problem as the reoptimization of the RPP. We first consider the case where a new required edge is added. Second, we address the case where an edge (required or not) is removed. We show that the reoptimization problems are $\mathcal{NP}$-hard. We consider a heuristic for the case where a new required edge is added which is a modification of the cheapest insertion algorithm for the traveling salesman problem and show that it has a worst-case ratio equal to 2. Moreover, we show that simple algorithms to remove an edge from an optimal RPP tour guarantee a tight ratio equal to 3/2. Computational tests are made to compare the performance of these algorithms with respect to the Frederickson algorithm running from scratch.

1. Introduction

Arc routing problems are vehicle routing problems where customers are represented by edges or arcs of a network. Such problems arise in a variety of practical contexts, such as street sweeping, garbage collection, mail delivery, meter reading or full truckload transportation. Even if they are much less studied than node routing problems, they recently inspired a rich and ever growing body of literature. We refer to the book edited by Dror [17] and the recent paper by Corberán and Prins [14] for a comprehensive overview on arc routing problems.

In this paper we consider the undirected Rural Postman Problem (RPP) which consists in determining a minimum cost cycle traversing at least once each edge belonging to a given subset of the edges of an undirected graph. The edges to be traversed are said to be *required*. The RPP has been introduced by Orloff [27] and later proved to be $\mathcal{NP}$-hard by Lenstra and Rinnooy Kan [25]. However, the special case of the RPP where the subgraph induced by the required edges is connected can be solved in polynomial time as it reduces to the Chinese Postman Problem (CPP) (e.g., see [18]).

The heuristic proposed by Frederickson [19] is the constructive algorithm with the best known approximation ratio for the RPP. The heuristic is designed along the lines of the algorithm proposed by Christofides [15] for the Traveling Salesman Problem (TSP) and, similarly to the latter, it has a worst-case ratio of 3/2.

In classical optimization is implicitly assumed that all the information concerning the instance is known at the time the search for an optimal solution begins. Additionally, it is also implicitly assumed that no information is available about the solution of the instance or of related instances. Even if this is the situation in several cases, it is often convenient to build partial solutions before all the information about the instance becomes known, or it frequently happens that new information is unexpectedly received after an optimal solution has been constructed, possibly with a heavy computational burden. For instance, in many applications it is convenient to find an optimal solution of an instance composed of a restricted set of inputs representing a constant over time environment and then managing dynamically the non-constant inputs. On the other hand, in important application areas, such as airlines [13], railways [33] and dial-a-ride problems [6,7], events that modify the instance before the solution is fully or partially implemented are frequent. The occurrence of these events is usually referred to as disruption and some examples are cancellation of orders, the break-down of a link in a communication network or the unavailability of one member of a crew. In all the former situations some questions arise: How long will it take to recover optimality? If the time taken is unacceptable, how can a good solution be obtained? Can one take advantage of the optimal solution already computed or should a new solution be computed from scratch?

The previous situations can be formalized as follows: Given an instance of an $\mathcal{NP}$-hard optimization problem together with an optimal solution, a new optimal solution has to be found after a perturbation of the instance has occurred. The problem of taking into account the knowledge of an optimal solution to solve the new instance is referred to as *reoptimization*. Reoptimization aims at determining whether the knowledge of an optimal solution of

* Corresponding author. Tel.: +39 030 2988588; fax: +39 030 2400925.
E-mail addresses: archetti@eco.unibs.it (C. Archetti), guastaro@eco.unibs.it (G. Guastaroba), speranza@eco.unibs.it (M.G. Speranza).

0305-0548/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.cor.2012.12.010

the original instance can be used to design either algorithms that achieve better approximation ratios than algorithms running from scratch or faster algorithms (or even both). In recent years, reoptimization problems are receiving increasing attention. Reoptimization has been studied for scheduling problems (e.g., [32,31]), for the 0–1 knapsack problem by Archetti et al. [2] and for set covering problems by Bilò et al. [8] and Mikhailyuk [26]. Several contributions appeared on reoptimization of graph-based problems. Gallo [22] and Pallottino and Scutellà [28] consider reoptimizing shortest paths when the cost of the arcs changes. Reoptimization for the minimum spanning tree problem is investigated by Boria and Paschos [12] and by Papadaki and Speranza [29]. In [29] the reoptimization of the maximum flow problem is also studied. The Steiner tree problem is analyzed in Böckenhauer et al. [10]. Several papers studying the reoptimization of variants of the TSP appeared in the literature. Archetti et al. [1] consider reoptimizing the TSP when a node is added or removed. Böckenhauer et al. [9,10] study the reoptimization problem when the weight of a single edge changes in a TSP instance. Ausiello et al. [3] investigate the (minimum) TSP and the maximum TSP when any number of nodes have to be inserted or removed from the instance. Böckenhauer and Komm [11] analyze the effectiveness of reoptimization of the TSP with deadlines, considering insertion and removal of a vertex as well as of a deadline. Finally, the recent contribution by Ausiello et al. [4] gives an overview on reoptimization problems. The authors first provide some general properties for the reoptimization of $\mathcal{NP}$-hard problems. Then, they review the application of reoptimization to some problems, namely the knapsack problem, the Steiner tree problem, a scheduling problem, and several variants of the TSP.

Contributions of the paper: We consider the following problem: Given an instance of the RPP together with an optimal solution, a perturbation of the instance takes place and a new instance has to be solved. We refer to this problem as the *reoptimization of the RPP*. We aim at exploiting the availability of an optimal solution of the original instance to find good solutions for the perturbed instance in an efficient manner, rather than running the optimization on the new instance from scratch. We consider two kinds of perturbations of the instance, namely the addition and the removal of one edge. In the case of one edge removal we address two cases separately, i.e. the case the edge is required and the case it is not. We analyze the computational complexity of the reoptimization problems. We prove that the considered reoptimization problems are $\mathcal{NP}$-hard. We propose simple approximation algorithms to solve them. We analyze the worst-case behavior of the approximation algorithms and show that they outperform in terms of running time other approximation algorithms for solving the RPP from scratch. Specifically, we show that a cheapest insertion algorithm has a worst-case ratio equal to 2 when used to insert the new edge into an optimal RPP tour. While the worst-case behavior is worse than the Frederickson algorithm [19], the computing time is linear instead of exponential in the number of connected components in the subgraph induced by the required edges [16]. Finally, we conjecture that the worst-case ratio is actually 3/2 and provide instance-related conditions under which the latter bound is tight.

We also introduce simple algorithms based on computing the shortest path between two nodes to tackle the problem of removing an edge from an RPP instance. The time complexity of the algorithms is polynomial, due to the computation of the shortest path (see [20]), and outperforms the Frederickson algorithm [19]. Moreover, we show that the approximation algorithms proposed to remove an edge guarantee a tight worst-case ratio equal to 3/2.

We perform computational experiments adapting benchmark RPP instances to the reoptimization problems. Computational results show the nodal importance of reoptimization both in terms of quality of the solution and efficiency of the computation.

Structure of the paper: The reoptimization problems are described in Section 2. The problems of inserting to and removing an edge from an optimal RPP tour are analyzed in Sections 3 and 4, respectively. Computational tests and results are illustrated in Section 5. Finally, in Section 6 some concluding remarks are drawn.

2. The reoptimization problems

We consider the undirected Rural Postman Problem (RPP) defined on a complete undirected graph $G=(V,E)$, where $V=\{1,2,\ldots,n\}$ is the set of nodes, E is the set of edges, and $d(i,j)$ is the symmetric distance between nodes i and j. A subset of edges $R\subset E$ is given, and each edge $(i,j)\in R$ is said to be *required*. The RPP is the problem of finding a minimum cost circuit traversing at least once all edges in R. The remaining edges, i.e. edges in $E\backslash R$, may be part of the solution. Hereafter, we will denote as $m=|R|$ the number of required edges. Moreover, we will denote as z^*_m the cost of an optimal tour for the RPP with m required edges. The optimal RPP tour is denoted as $t^*=(v_1,v_2,\ldots,v_{|t^*|})$, where v_b stands for the b-th node visited in tour t^*. Obviously, v_1 must coincide with $v_{|t^*|}$. We assume that distances satisfy the triangle inequality.

RPP$^+$ is the reoptimization problem when a new required edge is added to the instance. RPP$^+$ can be formally described as follows. Given an optimal solution of the RPP on G with m required edges, a new required edge $(n+1,n+2)$ and the distances $d(n+1,n+2)$, $d(i,n+1)$ and $d(i,n+2)$, with $i=1,\ldots,n$, determine an optimal tour for the RPP on the complete graph $G'=(V',E')$, where $V'=V\cup\{n+1,n+2\}$ is the set of nodes, E' is the corresponding set of edges, and $R'=R\cup\{(n+1,n+2)\}$ is the new set of required edges. The minimum cost of the RPP$^+$ is denoted as z^*_{m+1}. Note that an optimal tour for the RPP$^+$ is an optimal tour for the RPP with $m+1$ required edges.

The problem of reoptimizing an optimal tour for the RPP when an edge is removed from the instance is tackled by addressing two cases separately.

RPP^-_R is the reoptimization problem when an edge is removed and the edge removed is required. RPP^-_R can be formally defined as follows. Given an optimal solution of the RPP on G with m required edges and a required edge $(k,h)\in R$, determine an optimal tour for the RPP on graph $G=(V,E)$, where $R\backslash\{(k,h)\}$ is the new set of required edges. The minimum cost of the RPP^-_R is denoted as z^*_{m-1}. Note that an optimal tour for the RPP^-_R is an optimal tour for the RPP with $m-1$ required edges.

RPP^-_N is the reoptimization problem when an edge is removed and the edge removed is not required. We assume that the optimal RPP tour before the removal traverses the edge to be removed, otherwise no reoptimization is needed. RPP^-_N can be formally defined as follows. Given an optimal solution of the RPP on G with m required edges and a not required edge $(f,l)\in E\backslash R$, determine an optimal tour for the RPP on graph $G''=(V,E'')$, where $E''=E\backslash\{(f,l)\}$ is the new set of edges. The minimum cost of the RPP^-_N is denoted as $z^*_m(E'')$. Note that an optimal tour for the RPP^-_N is an optimal tour for the RPP with m required edges on graph G''.

3. Adding a new required edge to an optimal solution

In this section we study the RPP$^+$ problem. First, we show that no polynomial time algorithm exists for the problem unless $\mathcal{P}=\mathcal{NP}$. Then, we provide the worst-case analysis of a simple heuristic that inserts the new required edge into an existing optimal solution of the RPP.

We assume that even after adding the new required edge the subgraph induced by the required edges is still not connected otherwise the problem reduces to a CPP. Furthermore, note that