Solution Methods for the Multi-trip Elementary Shortest Path Problem with Resource Constraints

Zeliha Akca [*] Ted K. Ralphs [†] Rosemary T. Berger [‡]

December 31, 2010

Abstract

We investigate the *multi-trip elementary shortest path problem* (MESPPRC) with resource constraints in which the objective is to find a shortest path between a source node and a sink node such that nodes other than the specified *replenishment node* are visited at most once and resource constraints are not violated. After each visit to the replenishment node, which we take to be the sink node in our study, resource consumption levels can be reset to a certain initial value. As this problem arises primarily as a subproblem in decomposition-based algorithms for a wide variety of practical applications, we illustrate it in the context of an integrated routing and scheduling problem with capacitated and time restricted vehicles. We propose exact and heuristic algorithms and evaluate the performance of these in a branch-and-price framework.

1 Introduction

In this paper, we introduce a generalization of the well-known elementary shortest path problem with resource constraints (ESPPRC) that we call the *multi-trip ESPPRC* (MESPPRC). In the ESPPRC, the objective is to find a shortest path through a network between a given source node and a given sink node such that each other node is visited at most once (i.e., the path is *elementary*) and the resource constraints are not violated. In the MESSPRC, we look for a shortest path that is not technically elementary, but in which we can revisit the replenishment node (which we take to be the sink node in what follows) multiple times. In this generalization, the resource consumption of the paths may be reset to a lower value (often zero) when reaching the replenishment node.

The ESPPRC and its variants arise as substructures in many other models. For such models, decomposition-based solution algorithms, such as branch-and-price, are a natural approach. In branch-and-price, column generation is used to solve the LP relaxations, which have a large set of columns that are defined to be the solutions to a so-called *pricing problem (or subproblem)*. In each iteration of the column generation algorithm, columns that may improve the objective function (i.e., have negative reduced cost, assuming the objective is to minimize) are dynamically generated and added to the LP relaxation until no such columns remain. Because the pricing problem is solved repeatedly in this framework, it is crucial that the solution algorithm for this subproblem be extremely efficient.

[*]Basaksehir, Istanbul, Turkey zelihaakca@gmail.com

[†]Department of Industrial and Systems Engineering, Lehigh University, Bethlehem, PA 18015, USA ted@lehigh.edu, http://coral.ie.lehigh.edu/~ted

[‡]154 Gibbs Street, Unit 445 Rockville, MD 20850, USA rosemary.t.berger@gmail.com

Previous Work. Because elementary shortest path problems arise as the pricing problem in a wide variety of applications, a great deal of attention has been paid to solution of these problems in the literature. Examples of such applications include various vehicle routing and crew scheduling problems (Fukasawa et al. [2006], Kohl and Madsen [1997], Desrochers et al. [1992]), as well as integrated location and routing problems (Berger et al. [2007], Akca et al. [2009]). Depending on the structure of the particular problem being considered, the constrained shortest paths may represent, for example, vehicle routes or crew schedules. In the pricing problem, the costs of these paths are evaluated according to the "reduced cost" of the associated columns. The resource constraints on the paths come from associated constraints in the original problem. The most common resources considered are such things as vehicle capacity, time windows, and maximum path length. The motivation for studying the MESPPRC is that in some cases, we wish to allow paths that return multiple times to a replenishment point during a single "shift."

The ESPPRC is itself a generalization of the shortest path problem with resource constraints (SPPRC) in which the optimal path may visit a node multiple times. The standard exact solution procedure for the SPPRC is based on the dynamic programming approach proposed by Desrochers [1988]. He develops a labeling algorithm that has a pseudo-polynomial complexity depending on the size of the resource windows and the number of resources. The algorithm has been shown to be successful when there are tight resource constraints, but it becomes more time-consuming as the number of resources increases. As the algorithm builds partial paths, it assigns a label to each, indicating the resource consumption. To eliminate the problem of an increasing number of labels, the algorithm applies "dominance rules" to delete dominated labels. Irnich and Desaulniers [2005] provide a review of SPPRC, its variants, and its application as a pricing problem.

Beasley and Christofides [1989] extend SPPRC to ESPPRC and describe how to find elementary paths in a graph by adding an extra binary resource for each node indicating whether the node has been visited or not. However, this increases the number of resources and possible combinations exponentially. In fact, the ESPPRC is proved to be NP-Hard by Dror [1994]. When the underlying network of ESPPRC is acyclic, the elementary property of the paths can be ignored because any optimal solution of the SPPRC is guaranteed to be elementary in this case. When the network may have cycles and negative arc costs, which is generally the case in pricing problems, the elementary property of the paths must be considered.

To accelerate the computation time for the SPPRC algorithm, Dumitrescu [2002] proposes some preprocessing and bounding approaches which may also be applied to ESPPRC. Feillet et al. [2004] adapt Desrochers's labeling algorithm to find elementary paths and strengthen the algorithm by developing additional domination rules for the labels. Righini and Salani [2006] propose bounding approaches and also extend Feillet et al. [2004]'s algorithm to a bi-directional labeling algorithm by generating forward paths from source and backward paths from sink simultaneously. For solving the ESPPRC more efficiently, Boland et al. [2006] propose an extension to Desrochers's algorithm using the resource variables to enforce the elementary property for each node. They start solving the SPPRC and force the elementary property only for an iteratively augmented subset of nodes. As an alternative to labeling based approaches, Jepsen et al. [2008] present a mathematical programming model of ESPPRC and investigate valid inequalities for the problem. They introduce a new family of inequalities and develop a branch-and-cut algorithm for the problem that outperforms the labeling algorithm applied to the same problem.

Outline. One typical setting in which an algorithm for the ESPPRC is required is the solution of the capacitated vehicle routing problem by branch-and-price. Such a branch-and-price algorithm is typically based on either a set partitioning or set covering model in which the columns are associated with the vehicle routes. Here, the pricing problem is an ESPPRC in which the resource is the vehicle capacity and the goal is to generate the resource feasible paths with smallest reduced cost. In this paper, we focus on developing solution procedures for a pricing problem that generates *multi-trip vehicle routes*, thereby allowing integration

of routing and scheduling. Whereas in the classical vehicle routing problem, there is a one-to-one relationship between vehicles and routes, the integrated vehicle routing and scheduling problem adds scheduling constraints and seeks to determine the set of vehicle routes as well as the assignment of (possibly multiple) routes to each available vehicle. Taillard et al. [1996], Brandao and Mercer [1997], Petch and Salhi [2004], Olivera and Viera [2007] are some of the studies addressing integrated routing and scheduling problems using various heuristic approaches. Akca [2009] integrates the strategic problem of determining facility locations with the vehicle routing and scheduling decisions and provides exact solution approaches which require solution of MESPPRC as a pricing problem.

In the remainder of the paper, we formally define the MESPPRC and develop solution algorithms adapted from Feillet et al. [2004]'s labeling algorithm. We are able to solve the problem to optimality for cyclic networks and computationally evaluate the proposed approaches. Section 2 provides a formal description of the pricing problem utilizing MESPPRC and the description of the MESPPRC including a network based model and a mathematical programming formulation. Section 3 describes three exact algorithms for solving MESPPRC and Section 4 provides two heuristic extensions of these exact approaches. Section 5 provides algorithm-specific details of the computational experiments and presents computational results evaluating the performance of the algorithms in different settings. Section 6 concludes the paper.

2 The Multi-trip ESPPRC

To motivate the importance of the MESPPRC and to make the problem more concrete, we introduce it in the context of an integrated routing and scheduling application. Given a facility location and a set of customers that are candidates to be served from that facility, the MESPPRC is the problem of determining a subset of customers that can be visited by a set of vehicle routes assigned to a single vehicle, subject to scheduling (time) constraints. The objective of this pricing problem is to generate such a set of routes, called a *pairing*, with minimum cost. In this setting, the cost of a pairing is the reduced cost of the column associated with the schedule. This cost is additive with respect to the reduced costs of the individual arcs so that the objective function remains linear. In this study, we consider capacitated and time-limited vehicles.

To be more formal, we say that a pairing consists of (i) a set of customers, (ii) a partition of those customers into *routes*, and (iii) a permutation of the customers on each route indicating the order in which they should be visited. A pairing is feasible if for each route in the pairing, the total demand of the customers assigned to the route is at most the vehicle capacity and if the total travel time of all routes is at most the vehicle time limit. Figure 1 displays a visualization of a pricing problem and its solution. In the figure, the nodes are labeled with the demand associated with the customer represented by the node, while the travel time and cost of each arc is listed in parentheses next to the arc. The pairing in the example is composed of three routes, the total demand of each of which is less or equal to the vehicle capacity. The associated total travel time and cost are 60 and -17, respectively.

2.1 A Network Model

The above problem of generating the pairing of minimum cost can be interpreted as a generalization of an elementary shortest path problem with resource constraints (ESPPRC) on a network. To properly represent this problem as an MESPPRC, we need to define the network, determine the resources, and determine how to compute the resource consumption of the arcs so that a feasible path from the source node to the sink node corresponds to a feasible pairing as defined above.

As such, we construct an associated network with $|I|+2$ nodes, where I is the set of customer locations. In

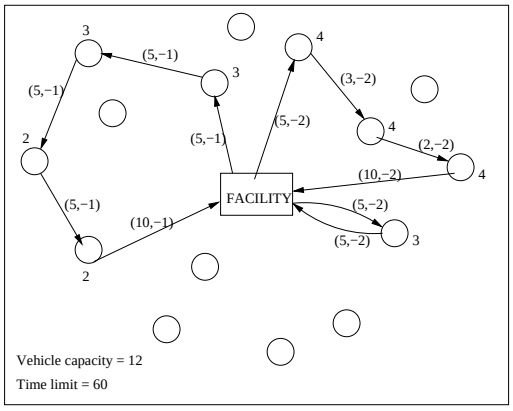


Figure 1: Problem layout and a possible solution (a pairing)

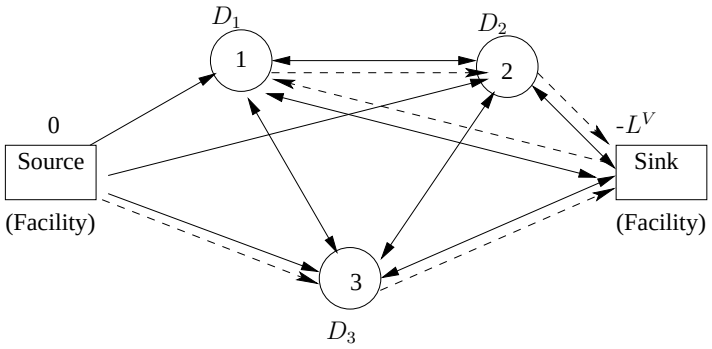


Figure 2: Example of a network and a path from source to sink (dashed line)

this network, we have one node associated with each customer, a node for the facility as the source node, and a copy of the facility as the sink node. The set B of arcs in the network consists of arcs from the source to customer nodes, arcs between customer nodes and the sink node, and arcs between pairs of customer nodes. Note that there are no arcs from customer nodes back to the source but there are arcs from the sink to customer nodes, enabling the generation of pairings including multiple routes. In a path in this network originating at the source node, an arc from the sink node to a customer node represents the beginning of a new route, whereas an arc from a customer node to the sink node represents the end of a route.

Let c_{ik} be the cost of arc (i,k) and T_{ik} be the travel time on arc (i,k) for all $(i,k) \in B$. Let D_i be the demand of customer i, $\forall i \in I$. L^V and L^T are the parameters that are equal to the capacity and the time limit of the vehicle, respectively. Figure 2 shows an example network and a path on the network. In the example, the path corresponds to pairing (facility - 3 - facility - 1 - 2 - facility), which includes two routes. We set the length and the cost of arc (i,k) to T_{ik} and c_{ik}, respectively, for all $(i,k) \in B$. We set the demand of each customer node i to D_i, while the source is assigned a demand of zero and the sink is (conceptually) assigned a demand of minus vehicle capacity demand ($-L^V$). We use time and vehicle capacity as the two resources in the problem. As we construct a path, when we include an arc, the length of the arc and the demand of the destination node are used to update the elapsed time and the vehicle load (resources consumed), respectively. In this network, an elementary resource constrained path from the source node to the sink node possibly visiting the sink node multiple times corresponds to a feasible pairing.

2.2 Mathematical Programming Formulation

A mathematical programming formulation of MESPPRC defined over the graph $G = (M, B)$, where M is the set of all nodes—the source node (denoted as s), the sink node (denoted as f), and those associated with customers—is as follows:

$$\text{Min} \quad \sum_{(i,k)\in B} c_{ik} x_{ik} \tag{1}$$

$$\text{s.t.} \quad \sum_{k\in M} x_{ik} \leq 1 \qquad \forall i \in I, \tag{2}$$

$$\sum_{k\in M} x_{ik} - \sum_{k\in M} x_{ki} = 0 \qquad \forall i \in I, \tag{3}$$

$$\sum_{k\in M} x_{sk} = 1, \tag{4}$$

$$\sum_{k\in M} x_{kf} - \sum_{k\in M} x_{fk} = 1, \tag{5}$$

$$y_{ik} - L^V x_{ik} \leq 0 \qquad \forall (i,k) \in B, \tag{6}$$

$$\sum_{k\in M} y_{ik} - \sum_{k\in M} y_{ki} + D_i \sum_{k\in M} x_{ik} = 0 \qquad \forall i \in I, \tag{7}$$

$$\sum_{(i,k)\in B} T_{ik} x_{ik} \leq L^T, \tag{8}$$

$$x_{ik} \in \{0,1\} \quad \forall (i,k) \in B, \tag{9}$$

$$y_{ik} \geq 0 \qquad \forall (i,k) \in B, \tag{10}$$

where $x_{ik} = 1$ if arc (i,k) is used in the solution and 0 otherwise; and y_{ik} is equal to the amount of flow on arc (i,k) for all $(i,k) \in B$.

The objective function (1) indicates that the total cost, which is the sum of the arc costs included in the pairing, should be minimized. Constraints (2) specify that the customer node i may be visited at most once. Constraints (3) require that the vehicle should enter and leave a customer node an equal number of times. Constraint (4) defines that the path starts at the source node. Constraint (5) specifies that the sink node may be visited multiple times and the path ends at the sink node. Constraints (6) guarantee that the vehicle capacity is not exceeded. Constraints (7) require conservation of flow at each customer node. Constraints (8) limit the total time of a vehicle's schedule to the time limit. Constraints (9) and (10) are the integrality and non-negativity requirements on the variables.

3 Exact Algorithms

One approach to solve MESPPRC is to use a branch-and-cut methodology based on the mathematical programming formulation presented in the previous section. Instead, based on our computational experience, we proceed with network based labeling algorithms. Because the graph G may be cyclic, we have to consider the elementary property of the paths in order to generate a feasible pairing. We use two extensions of Feillet et al. [2004]'s algorithm for the ESPPRC. In the first, we directly modify the algorithm to account for the presence of the replenishment node and apply it to the network described in Section 2.1. In the second, we extend the first approach to a two-phase algorithm in which each phase involves solving a modified ESPPRC by Feillet et al. [2004]'s original algorithm.

3.1 One-Phase Algorithm

Basic Algorithm. In this first approach, we adapt Feillet et al.'s algorithm in order to solve MESPPRC on the constructed network. In the Feillet et al. algorithm, each path from the source to a node in the network is assigned a label, which is a vector of (1) the cost of that (partial) path, (2) the resources consumed so far (vehicle load and elapsed time), (3) a resource corresponding to each customer node indicating whether the customer has already been visited or not, and (4) the total number of unreachable customers. Customers are considered *unreachable* if they have already been visited in the path or if they cannot be served by the vehicle without violating a resource constraint. The algorithm fans out from the source node, repeatedly examining the nodes of the graph. Each time it examines a node, the algorithm extends each of the paths to each possible successor node, continuing until no further extensions are possible.

To define notation, we let l_r^i be a label that corresponds to path r from source to node i. We use vector $l_r^i = (R_c^i, R_v^i, R_t^i, R_n^i, V_1^i, .., V_{|I|}^i)$ to store the information about path r where

R_c^i = cost of the (partial) path,

R_v^i = total demand of the (partial) path,

R_t^i = time consumption of the (partial) path,

R_n^i = total number of unreachable customer nodes (does not include temporary unreachable nodes) (so far),

$$V_k^i = \left\{ \begin{array}{ll} -2 & \text{if customer } k \text{ is temporarily unreachable for} \\ & \text{the (partial) path (since vehicle capacity constraint)} \\ -1 & \text{if customer } k \text{ is unreachable for the (partial) path} \\ & \text{because of the resource limits,} \\ 0 & \text{if customer } k \text{ is not in the (partial) path and it is} \\ & \text{reachable for the path with the remaining resources,} \\ s > 0 & \text{if customer } k \text{ is the } s^{th} \text{ node in the (partial) path,} \end{array} \right\} \forall k \in I.$$

With respect to label l_r^i, if either customer k has already been visited or visiting customer k is resource infeasible with respect to time constraint, node k is considered unreachable (i.e., $V_k^i = s$ or -1). Even though we have two resource constraints (vehicle capacity and time limit), the time limit constraint is the only one checked to mark a customer as unreachable because if the vehicle load exceeds vehicle capacity by adding customer k to the path, that does not mean that we cannot add customer k to this path in the future. It may be possible that a path extended from path r stops at the sink node and would have enough capacity to visit customer k in further extensions. Therefore, we mark the node as temporarily unreachable (i.e., $V_k^i = -2$), if it is resource infeasible with respect to vehicle capacity.

Extending a path to the sink is always resource feasible because (1) demand of the sink is the negation of vehicle capacity (and hence total demand would be less than the vehicle capacity), and (2) while constructing label r associated with a non sink node i, the time limit constraint is always checked to ensure that the path is extendable to the sink.

Our algorithm for solving the ESPPRC is referred to as Algorithm 1P and includes three main steps:

Algorithm 1P:

1. Initialization step: A label list is created for each node to keep the generated labels. All label lists except the list for the source node are initially empty. The source has one label, which is initialized as a zero vector.

2. Extension step: For each node, the label list is searched and all unprocessed labels (those not extended in previous iterations) are extended to all possible successors of the node. The label l_r^i for node i can be extended to node k if (i) k is the sink node or $V_k^i = 0$ and (ii) the path is resource feasible, i.e., $R_v^i + D_k \leq L^V$

and $R_t^i + T_{i,k} + T_{k,sink} \leq L^T$. If it is feasible to extend l_r^i to node k, a new label is created for node k by updating the resource consumptions and cost. Mark the labels that were extended as processed.

3. Elimination step: Newly constructed labels are combined with the previously created label lists and only the *non-dominated labels* are retained. Let l_1 and l_2 be two labels. Based on Feillet et al., label l_1 dominates label l_2 if all of the following conditions are satisfied:

i. the current travel time, the vehicle load, and cost of l_1 are less than or equal to those of l_2,

ii. the number of total unreachable customer nodes of l_1 are less than or equal to those of l_2, and for all nodes $g \in I$, g is reachable only in label l_1 or g is unreachable in both labels l_1 and l_2.

All dominated labels are deleted from the label lists of the nodes. The algorithm returns to the extension step until all labels are processed and no new labels are added.

Extended Algorithm. We further extend Feillet et al.'s algorithm by adding a new domination rule. Executing the described Algorithm 1P on the constructed network leads to paths from the source to the sink that correspond to sets of *ordered* vehicle routes. Because the ordering of the routes within a pairing is arbitrary for our problem, we would like to avoid generating multiple paths that are different permutations of a set of routes and correspond to the same pairing. For example, a call to Algorithm 1P may generate the following paths:

(source - 1 - 2 - sink - 5 - 3 - 4 - sink - 6 - sink), (source - 1 - 2 - sink - 6 - sink - 5 - 3 - 4 - sink),

(source - 5 - 3 - 4 - sink - 1 - 2 - sink - 6 - sink), (source - 5 - 3 - 4 - sink - 6 - sink - 1 - 2 - sink),

(source - 6 - sink - 1 - 2 - sink - 5 - 3 - 4 - sink), and (source - 6 - sink - 5 - 3 - 4 - sink - 1 - 2 - sink).

However, these six paths are equivalent with respect to total cost and time, and they correspond to the same pairing composed of the following three routes: (facility - 1 - 2 - facility), (facility - 5 - 3 - 4 - facility) and (facility - 6 - facility).

To eliminate generation of all possible ordered sets, we introduce a new domination rule in which we force a specific order for routes in the pairing that decreases the state space of the algorithm significantly, as follows.

Rule The index of the first customer node of each route should be less than the index of the first customer node in the following route.

The modified algorithm only generates paths from source to sink in which the first node of each route is less than the first node of the following route. Thus, the algorithm with this rule generates only path (source - 1 - 2 - sink - 5 - 3 - 4 - sink - 6 - sink) in the example above.

To apply this rule, we add variable R_f^i to keep the index of the first customer node in the current route in the path to the vector for label l_r^i. We make the following modifications in Algorithm 1P:

- *In the extension step of the algorithm*, while constructing label l_p^k from l_r^i, we set $R_f^k = k$ if i is the source or the sink (k is the first node in the new route), otherwise, R_f^k equals to the first node of the extended label. If k is the sink, for all reachable node $g \in I$ such that $g < R_f^k$, we also set $V_g^k = -2$ (temporarily unreachable). Currently, such a node cannot be the first node in the new route. However, it may still become part of the path during further extensions. In addition, while extending the path to nodes other than the first node in the route, we change the "temporarily unreachable" condition to "reachable."

- The modification to the *elimination step of the algorithm* is to keep the label with smallest first node in the current route, in case each label of the compared pair of labels dominates the other.

We refer to this extended algorithm as Algorithm 1P*.

3.2 Two-Phase Algorithm

An alternative for solving the MESPPRC exactly is to extend the one-phase approach to two phases: first, generate a set of feasible vehicle routes and then combine them to construct a feasible pairing of minimum cost. Each phase is formulated as a network problem and solved as an ESPPRC. Since a pairing is a set of routes, we can express the variable part of the cost of a pairing as the sum of the costs of the routes forming the pairing. For there to exist a negative cost pairing composed of more than one route, there must exist routes with negative cost. Otherwise, the pairing including only the single route with the lowest cost would be the lowest cost pairing overall.

Phase One: Generating Routes. To generate only feasible routes instead of pairings, we solve a regular ESPPRC problem on the constructed network with $|I|+2$ nodes. To do so, we delete the arcs from the sink to customers and use Feillet et al.'s algorithm with vehicle capacity and time limit resources. Each label at the sink node then corresponds to a feasible vehicle route. Let L_F denote the set of feasible routes with negative cost at the sink. If L_F is non-empty, we continue with phase two.

Phase Two: Generating Pairings. The objective of phase two is to generate feasible pairings with negative cost by combining the routes in set L_F. Two (or more) vehicle routes can be combined to form a feasible pairing if they do not have customer nodes in common and they can both (all) be completed within the vehicle's time limit. Finding a feasible pairing with the minimum cost again corresponds to solving an ESPPRC with a single resource constraint (time).

We construct a network with $2+|L_F|$ nodes: a source node, a sink node and one *intermediate* node to represent each of the vehicle routes in set L_F. The network includes an arc from the source node to each intermediate node, arcs between pairs of intermediate nodes, and an arc from each intermediate node to the sink node. Note that, because the sequence of routes in a pairing is arbitrary, we only include an arc from an intermediate node to another intermediate node with higher index. Associated with each arc inbound to an intermediate node are two values: the time required to complete the corresponding route and the cost of the route. The time and cost for arcs inbound to the sink are zero. To find a feasible pairing with the minimum cost, we directly apply Feillet et al.'s algorithm. The constructed network is acyclic, so we do not have to keep track of the elementary property of the path in this network. However, we still consider the elementary property with respect to the customer nodes by comparing the resource associated with each customer. Any pairing with the smallest cost is a candidate pairing to be the optimal solution.

Since we solve the exact ESPPRC algorithm in each phase of the algorithm, the overall pricing algorithm is exact and results in a pairing with the smallest cost. We refer to this exact pricing algorithm as Algorithm 2P. To clarify the algorithm, we demonstrate how we construct the networks for each phase for an example in Figure 3.

3.3 Algorithmic Extensions

We have computationally seen that the following extensions to the algorithms proposed significantly improve the performance of the algorithms.

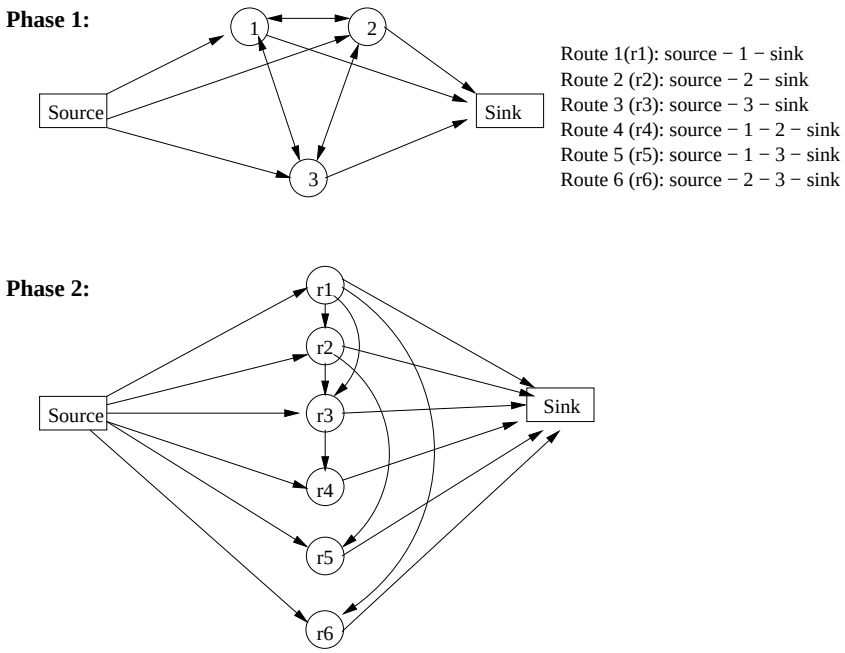


Figure 3: Demonstration of Phase 1 and Phase 2 Networks on a Small Example

- In general, at any iteration, there are multiple non-dominated labels representing partial paths at each node of the network. Sorting the label list of each node in ascending order of cost improves the performance of the algorithm by eliminating more labels via domination rules. This approach is commonly used in the literature.
- In the extension steps of the algorithms, we select a node among all nodes for which the label list is non-empty. Instead of searching a node in an ordered list of nodes from 1 to $|I|+2$, we choose the node randomly. This leads to more and faster expansion of the label lists of the nodes and improves the performance of the algorithms.

4 Heuristic Algorithms

To find feasible pairings more quickly, here we define two heuristic versions of the multi-trip shortest path algorithms. We can use these heuristic modifications for any of Algorithms 1P, 1P*, or 2P.

4.1 Label Limit

This heuristic version was first proposed by Dumitrescu [2002] and restricts the state space of the exact algorithm by directly restricting the number of labels. In particular, during Algorithms 1P or 1P* or either phase of Algorithm 2P, we limit the maximum number of non-processed labels allowed at each node. During the algorithm, as we update the list of labels in the elimination step, we order the non-dominated labels in *increasing order of cost* and we keep at most the first *label limit* number of non-processed labels. For small label limits, the algorithm terminates very quickly. As the label limit gets larger, the heuristic version approaches to the exact algorithm and the quality of the solution improves. In order to trade speed against solution quality effectively, we implement this heuristic by gradually increasing the label limit.

4.2 Restricted Customer Set

This second heuristic restricts the state space by considering only a subset of the customers. In this way, the total number of labels to be generated is restricted in advance. First, we determine the *two customers* that have the *lowest arc costs from the facility*.

We let j be the facility index, and choose customers i_1 and i_2 such that $c_{ji_1} \leq c_{ji_2} \leq c_{ji}$ for all $i \in I \backslash \{i_1, i_2\}$. We denote by I_u the subset of I in which we run the algorithm and initialize the subset with $I_u = \{i_1, i_2\}$. Then by exploring the arcs from I_u the customers in $I \setminus I_u$, we repeatedly add one customer that is connected by a cheapest arc to the subset until the size of the subset is equal to the determined *subset size*. In particular, we let $I_u = I_u \cup \{k\}$ where $k = \text{argmin}_{k \in I \setminus I_u}\{\min_{i \in I_u}\{c_{ik}\}\}$. We stop when $|I_u| = u$ where u is the determined subset size. After we determine I_u, we execute one of the exact MESPPRC algorithms but with this smaller customer set.

5 Computational Experiments

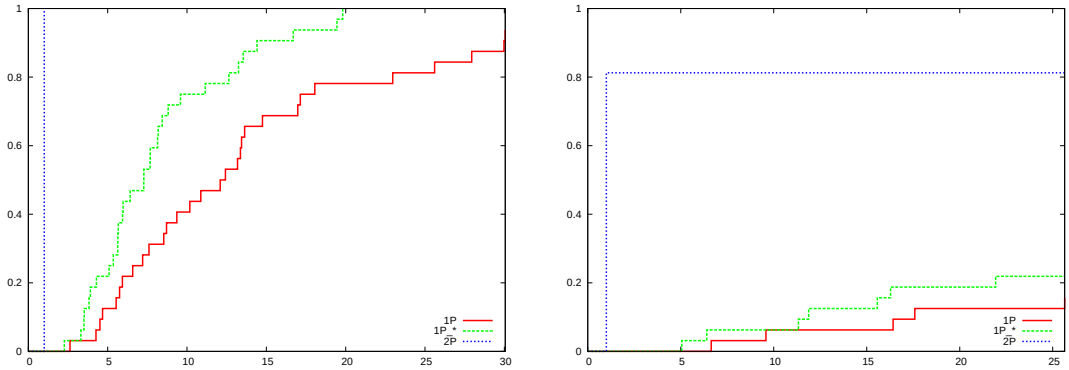
In this section, we present computational experiments to empirically validate the algorithms discussed and evaluate their performance of the proposed algorithms in a branch-and-price environment. In our experiments, we use the branch-and-price algorithm developed to solve an integrated location, routing and scheduling problem (LRSP) described in Akca [2009]. Given a set of candidate facility locations and a set of customer locations, the objective of the LRSP is to select a subset of the facilities, construct a set of delivery routes, and assign routes to vehicles considering capacity restricted facilities, and capacity and time restricted vehicles. The formulation of the problem includes the set partitioning constraints and columns corresponding to feasible pairings associated with a facility. Therefore, in this application, an MESPPRC must be solved for each facility in each iteration.

Akca [2009] describes 32 instances with 25 customers and 32 instances with 40 customers, which we also use for our testing here. All instances include five candidate facility locations. They are generated based on the instances presented by Cordeau et al. [1997] for multi-deport vehicle routing problem, in particular, instance group $p01$ (50 customers), $p03$ (75 customers), and $p07$ (100 customers) are selected to generate the LRSP instances. For each location set, two time limit values and two vehicle capacity values are considered. In the experiments, we use performance profiles (introduced by Dolan and More [2002]) to evaluate the performance of the algorithms.

5.1 Performance of Exact Algorithms

First, we compare the performance of the exact solution approaches: Algorithms 1P, 1P*, and 2P. To compare the performance of these algorithms, we solved the LP relaxation of the root node in the branch-and-price algorithm with column generation employing each one of these algorithms. In the experiments, at each iteration of the column generation algorithm, for each facility, at most one column (the one with the most negative cost) was generated. We ran the experiments with a time limit of 3 CPU hours and using the 25- and the 40-customer instances.

Figure 4 presents the performance profiles for the time required to solve the root node using each of the multi-trip shortest path algorithms. The results clearly show that Algorithm 2P significantly outperforms both Algorithms 1P and 1P*. The effect of employing Algorithm 2P is even more significant for the 40-customer instances. Within the time limit of 3 CPU hours, the column generation algorithm with Algorithms 1P and 1P* could solve the root node in fewer than 25% of the 40-customer instances, while Algorithm 2P



(a) 25-customer Instances (b) 40-customer Instances

Figure 4: Performance profiles for the root node solution time using Algorithms 1P, 1P*, and 2P

could solve 82% of the instances.

In our computational experiments, the average time per column generation iteration improved 43% with Algorithm 1P* and 94% with Algorithm 2P compared to Algorithm 1P in the 25-customer instances. In the 40-customer instances, the average time per column generation iteration improved 36% with Algorithm 1P* and 70% with Algorithm 2P (for tables displaying the results for each instance see Akca [2009]). In summary, Algorithm 2P performed significantly better than Algorithms 1P and 1P*.

5.2 Performance of Heuristic Algorithms

In this section, we empirically assess the effect of using heuristics to solve the pricing problem on the overall performance of the branch-and-price algorithm from the last section. In Section 4, we described the algorithms with label limit and for a restricted subset of the customers. Since our experiments proved that the exact Algorithm 2P is superior to other exact algorithms, heuristic versions of Algorithm 2P, which we refer to as Algorithm 2P-LL (algorithm with label limit) and Algorithm 2P-CS (algorithm with subset of customers) are evaluated in this section.

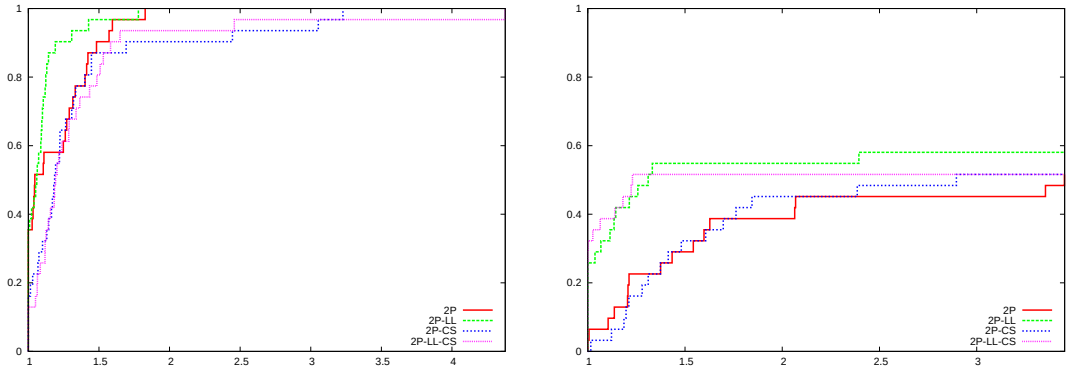
Performance within an Exact Branch-and-Price Algorithm. In the first set of experiments, we executed the branch-and-price algorithm using an exact column generation algorithm that employs Algorithms 2P, 2P-LL, and 2P-CS in combination. In each run, we set a time limit of 8 CPU hours. We compared four variants of the exact branch-and-price algorithm:

1. In the first variant, we used a column generation algorithm employing only exact pricing Algorithm 2P.

2. In the second variant, we used a column generation algorithm employing Algorithm 2P-LL before calling Algorithm 2P.

3. In the third variant, we used column generation algorithm employing Algorithm 2P-CS before calling Algorithm 2P.

4. Finally, we used both Algorithm 2P-CS and Algorithm 2P-LL before applying Algorithm 2P.



(a) 25-customer Instances (b) 40-customer Instances

Figure 5: Performance profiles for the total time of the branch-and-price employing only Algorithm 2P or performing one or two of Algorithm 2P-LL and Algorithm 2P-CS before Algorithm 2P

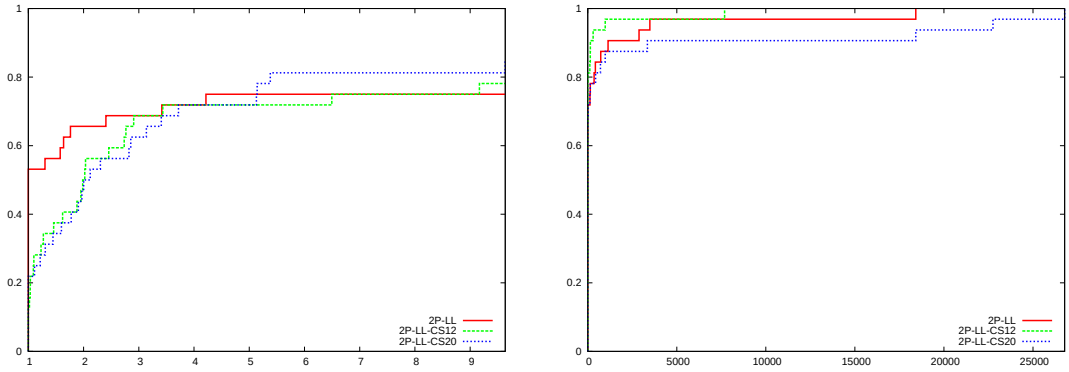
Implementation of Algorithm 2P-LL: To be able to improve the performance of the column generation algorithm, we start Algorithm 2P-LL with an initial label limit l and gradually increase it to an upper bound L. In particular, if $l < L$, we execute Algorithm 2P-LL with label limit l until the algorithm terminates with no new pairings. We then set $l = 2l$ and re-execute Algorithm 2P-LL. Based on preliminary computational experiments, we decided to set l= 50 and L = 350 in all experiments. In general, the number of columns with negative cost is larger in early iterations of the column generation algorithm. Therefore, the number of labels at each node in the pricing algorithm tends to increase rapidly in the early iterations. Increasing the label limit gradually, we aim to pass the earlier iterations of the column generation quickly, and explore a larger search space in later iterations in order to be able to generate a column with small cost.

Implementation of Algorithm 2P-CS: In Algorithm 2P-CS, we set a subset size u and determine a subset of customers as described in Section 4. Then we execute Algorithm 2P-CS at most U iterations or until the algorithm terminates with no new columns. After some preliminary experiments, we decided to set U = 12 in all experiments, with $u = 12$ for the 25-customer instances and $u = 15$ for the 40-customer instances. For large values of u, the heuristic becomes slower. However, if u is not large enough, the quality of the generated pairings may not be sufficient (i.e., it may include too few customers). Note that the value of u differs based on the number of customers in the instance. We chose the values of u by inspecting the solutions we found in initial testing. In addition, our preliminary experiments showed that U values greater than 20 reduced the overall performance of the algorithm.

Figure 5 presents the performance profiles for the total time spent in each variant of the exact branch-and-price: exact branch and price that uses only Algorithm 2P, exact branch-and-price that also calls Algorithm 2P-LL, exact branch-and-price that calls Algorithm 2P-CS before Algorithm 2P, and exact branch-and-price that calls both heuristics.

Based on Figure 5(a), for the 25-customer instances, the variant that employs Algorithm 2P-LL outperforms all others. The variant that uses only Algorithm 2P seems to be better than the ones that call only Algorithm 2P-CS or both Algorithms 2P-CS and 2P-LL. Therefore, we conclude that the use of Algorithm 2P-CS does not improve the total solution time of the algorithm for the 25-customer instances.

For the 40-customer instances, Figure 5(b) shows that the variants of the branch-and-price algorithm that employ Algorithm 2P-LL outperform those without Algorithm 2P-LL. The variant that calls both Algorithms 2P-LL and 2P-CS is better than the variant that calls only Algorithm 2P-LL in 55% of the instances. However, the variant that calls only Algorithm 2P-LL solves more instances within the time limit.



(a) Total time of the algorithm (b) (%Gap)×100

Figure 6: For 40-customer instances, performance profile for the time and the upper bound quality of the algorithm employing only Algorithm 2P-LL, and both Algorithms 2P-LL and 2P-CS with $U = 12$ or 20

Overall, the computational results show that the addition of Algorithm 2P-LL to the column generation procedure improves the performance of the branch and price algorithm. Even though the addition of Algorithm 2P-CS does not cause any positive improvement for the 25-customer instances, we observed that the variant of the branch-and-price algorithm that includes both Algorithms 2P-LL and 2P-CS performed well for the 40-customer instances. Therefore, we decided to investigate the performance of the Algorithm 2P-CS further in 40-customer instances.

Performance within a Heuristic Branch-and-Price Algorithm. Based on the results of our previous experiments, we decided to investigate the effect of Algorithm 2P-CS in the heuristic branch-and-price algorithm framework described in Akca [2009]. At each node of the heuristic branch and price tree, the heuristic column generation algorithm, in which only heuristic versions of the multi-trip shortest path algorithms, Algorithm 2P-CS and Algorithm 2P-LL are used. We constructed two variants of the algorithm:

1. In the first variant, we apply only Algorithm 2P-LL as a pricing algorithm.
2. In the second variant, we employ both Algorithm 2P-CS and Algorithm 2P-LL.

In the experiments, we set $l = 5$ and L=15, and $u = 15$. We varied the second design by using two different values (12 and 20) for U. Heuristic branch-and-price was run with a time limit of 2 CPU hours in each case.

Figure 6(a) compares the time spent in each variant of the algorithm. The figure shows that the algorithm with only Algorithm 2P-LL is better than the others in 70% of the instances. However, the variants that employ Algorithm 2P-CS solve more of the instances within the time limit.

We evaluate the quality of the upper bounds obtained in each heuristic branch-and-price algorithm in Figure 6(b). The performance profiles are based on the percentage gap between the upper bound at the end of the algorithm and the optimal or best known solution (IP). To generate the performance profiles, we replaced the 0% gaps with 0.0001% in order to avoid division by zero while calculating the ratio of performance measures. The algorithm using Algorithm 2P-CS with U = 12 outperforms the other variants in terms of the upper bound quality.

Overall, the computational results indicate that both heuristic extensions, Algorithms 2P-LL and 2P-CS improves the performance of the branch-and-price algorithm in terms of execution time and the upper bound quality.

6 Conclusions and Future Work

We have introduced a generalization of the ESPPRC that we refer to as the multi-trip ESPPRC. This problem allows multiple trips to a replenishment node at which the resource consumption of the path can be reset to zero. The problem may arise as a subproblem in the solution of various problems. In this study, as a particular example, we investigate the MESPPRC as a pricing problem in the solution of an integrated routing and scheduling problem for which MESPPRC generates vehicle schedules composed of vehicle routes that can be served within the time limit of the vehicles.

We have developed exact solution algorithms for MESPPRC based on Feillet et al. [2004]'s algorithm developed for ESPPRC. We have constructed a network for which solution of ESPPRC providing solutions to MESPPRC. We have further extended this approach with an additional domination rule significantly reducing the state space of the algorithm. Then, we have developed a two-phase solution methodology in which each phase has been defined as an ESPPRC. In order to improve the performance of the column generation algorithm that requires iterative solution of MESPPRC, we have described heuristic extensions of the solution algorithms for MESPPRC. We have tested the performance of the algorithms presented in the paper using variants of the applications in the branch-and-price environment. We have shown the effectiveness of the algorithms in various computational settings.

We are currently investigating a column generation methodology which utilizes solution of two subproblems iteratively. In this context, we are studying possible extensions or utilization of the two-phase solution approach proposed in this paper for this methodology. As a future study, we intend to develop additional heuristic extensions and some preprocessing phases for the algorithms for the MESPPRC. These extensions would lead to solution of very large integrated routing and scheduling problems using exact branch-and-price algorithm. In addition, we plan to test the performance of the algorithm with multiple replenishment points with multiple resources.

References

Z. Akca. *Integrated Location, Routing And Scheduling Problems: Models And Algorithms*. Phd thesis, Lehigh University, Bethlehem, PA, 2009.

Z. Akca, R. T. Berger, and T. K. Ralphs. A Branch-and-Price Algorithm for Combined Location and Routing Problems Under Capacity Restrictions. *The Proceedings of the Eleventh INFORMS Computing Society Meeting*, pages 309–330, 2009.

J. E. Beasley and N. Christofides. An Algorithm for the Resource Constrained Shortest Path Problem. *Networks*, 19:379–394, 1989.

R. T. Berger, C. R. Coullard, and M. S. Daskin. Location-Routing Problems with Distance Constraints. *Transportation Science*, 41:29–43, 2007.

N. Boland, J. Dethridge, and I. Dumitrescu. Accelerated Label Setting Algorithms for the Elementary Resource Constrained Shortest Path Problem. *Operations Research Letters*, 34:58–68, 2006.

J. Brandao and A. Mercer. A Tabu Search Algorithm for the Multi-Trip Vehicle Routing and Scheduling Problem. *European Journal of Operational Research*, 100:180–191, 1997.

J. F. Cordeau, M. Gendreau, and G. Laporte. A Tabu Search Heuristic for Periodic and Multi-Depot Vehicle Routing Problems. *Networks*, 30:105–119, 1997.

M. Desrochers. An Algorithm for the Shortest Path Problem with Resource Constraints. Technical Report G-88-27, GERAD, 1988.

M. Desrochers, J. Desrosiers, and M. M. Solomon. A New Optimization Algorithm for the Vehicle Routing Problem with Time Windows. *Operations Research*, 40(2):342–354, 1992.

E. D. Dolan and J. J. More. Benchmarking Optimization Software With Performance Profiles. *Mathematical Programming*, 91(2):201–213, 2002.

M. Dror. Note on the Complexity of the Shortest Path Models for Column Generation in VRPTW. *Operations Research*, 42(5):977–978, 1994.

I. Dumitrescu. *Constrained Path and Cycle Problems*. PhD thesis, Department of Mathematics and Statistics, University of Melbourne, 2002.

D. Feillet, P. Dejax, M. Gendreau, and C. Gueguen. An Exact Algorithm for the Elementary Shortest Path Problem with Resource Constraints: Application to Some Vehicle Routing Problems. *Networks*, 44(3): 216–229, 2004.

R. Fukasawa, H. Longo, J. Lysgaard, M. Poggi de Aragão, M. Reis, E. Uchoa, and R. F. Werneck. Robust Branch-and-Cut-and-Price for the Capacitated Vehicle Routing Problem. *Mathematical Programming*, 106(3):491–511, 2006.

S. Irnich and G. Desaulniers. *Column Generation*, chapter Shortest Path Problems with Resource Constraints, pages 33–65. GERAD 25th Anniversary Series. Springer, 2005.

M. Jepsen, B. Petersen, and S. Spoorendonk. A Branch-and-Cut Algorithm for the Elementary Shortest Path Problem with Capacity Constraint. *Technical Report, DIKU, University of Copenhagen, Denmark*, 2008.

N. Kohl and O. B. G. Madsen. An Optimization Algorithm for the Vehicle Routing Problem with Time Windows Based on Lagrangian Relaxation. *Operations Research*, 45(3):395–406, 1997.

A. Olivera and O. Viera. Adaptive Memory Programming for the Vehicle Routing Problem with Multiple Trips. *Computers and Operations Research*, 34:28–47, 2007.

R. J. Petch and S. Salhi. A Multi-Phase Constructive Heuristic for the Vehicle Routing Problem with Multiple Trips. *Discrete Applied Mathematics*, 133:69–92, 2004.

G. Righini and M. Salani. Symmetry Helps: Bounded Bi-directional Dynamic Programming for the Elementary Shortest Path Problem with Resource Constraints. *Discrete Optimization*, 3:255–273, 2006.

E. D. Taillard, G. Laporte, and M. Gendreau. Vehicle Routing with Multiple Use of Vehicles. *The Journal of Operational Research Society*, 47(8):1065–1070, 1996.