Models and Column Generation Approach for the Resource-constrained Minimum Cost Path Problem with Relays

Xiangyong Li, Shaochong Lin, Peng Tian, Y.P. Aneja

www.elsevier.com/locate/omega

PII: S0305-0483(16)00013-X
DOI: http://dx.doi.org/10.1016/j.omega.2016.01.012
Reference: OME1645

To appear in: *Omega*

Received date: 26 August 2015
Revised date: 24 November 2015
Accepted date: 27 January 2016

Cite this article as: Xiangyong Li, Shaochong Lin, Peng Tian and Y.P. Aneja, Models and Column Generation Approach for the Resource-constrained Minimum Cost Path Problem with Relays, *Omega*, http://dx.doi.org/10.1016/j.omega.2016.01.012

Models and Column Generation Approach for the Resource-constrained Minimum Cost Path Problem with Relays

Xiangyong Li
School of Economics & Management, Tongji University, Shanghai 200092, China, xyli@tongji.edu.cn

Shaochong Lin
School of Economics & Management, Tongji University, Shanghai 200092, China, 1991chong@tongji.edu.cn

Peng Tian
Antai College of Economics & Management, Shanghai Jiao Tong University, Shanghai 200052, China, ptian@sjtu.edu.cn

Y.P. Aneja
Odette School of Business, University of Windsor, Windsor, Ontario, N9B 3P4, Canada, aneja@uwindsor.ca

In this paper, we introduce the resource-constrained minimum cost path problem with relays (RMCPR), which deals with multiple-resource constraint on the path with relays. The RMCPR consists of finding a path from a source to a destination, and locating relays on some nodes of the path, such that the total cost of setting arcs and relays is minimized, and for each resource, the total consumption between the source and the first relay, any two consecutive relays, and the last relay and the destination do not exceed a predefined upper bound of resource consumption. The RMCPR is a single-commodity resource-constrained network design problem with relays.

We present a pattern-chain formulation and develop a column generation based exact approach for the RMCPR. We design a Lagrangian relaxation based method to efficiently price out columns to enter the basis. We present computational results on three sets of 560 randomly generated instances with different properties. Computational results demonstrate that our proposed algorithm is an efficient exact method for solving the RMCPR.

Key words: network design; relay location; multiple resource constraint; Lagrangian relaxation; column generation

1. Introduction

In this paper, we study the resource-constrained minimum cost path problem with relays (RMCPR), which can be used to model many network design problems. For example in freight transport systems, it becomes impractical for both truck drivers and the whole transportation process to cover very long-haul distance in one trip (Willoughby and Uyeno 2001). In the truckload industry, the long-distance travel leads to a serious and chronic

problem of high turnover rate, e.g., typically more than 100% (Üster and Kewcharoenwong 2011). This requires setting relay points along the paths for exchange of drivers, trucks and trailers. These relay points can be used for various purposes, e.g., driver rest and gasoline station. In addition, the RMCPR can also be used to model the network design problems in telecommunication systems, where the relay points may be used to boost signal quality over long distance (Cabral et al. 2007). More importantly, long-distance travel will inevitably lead to consumption of varieties of resources, typically including fuels of vehicle, time, distance, capacity, money, workload, and reliability requirements (Zhu and Wilhelm 2007). All these introduce issues of multi-resource constraints in the network design problems.

We now formally define the RMCPR over a directed network $G = (N, A, K)$ with node set $N = \{1, 2, \ldots, n\}$, arc set A ($|A| = m$), and resource set $K = \{1, 2, \ldots, r\}$. Each arc (i, j) has an installation cost of $c_{i,j}$ and an r-dimensional resource consumption vector $\boldsymbol{w_{i,j}} = (w^1_{i,j}, w^2_{i,j}, \ldots, w^r_{i,j})$, where $w^k_{i,j}$ represents the consumption of resource k along arc (i, j). A fixed cost g_i occurs when a relay is located at node i. The RMCPR consists of selecting network arcs, finding a path from the source s to the destination t, and locating relays on some nodes of the path, such that the total cost of setting arcs and relays is minimized, and in the meantime for each resource k, the total resource consumption along the subpath between the source and the first relay, any two consecutive relays, and the last relay and the destination do not exceed a predefined maximum resource consumption W^k. The RMCPR essentially can be viewed as a single-commodity multi-resource-constrained network design problem with relays.

To illustrate, we give an example of the RMCPR with three resources (e.g., $K = \{1, 2, 3\}$) in Figure 1. In this example, we need to determine a path from source s to destination t. The predefined maximum resource consumption W^k is 5 for each resource. A feasible path is given by the full line with relays located at nodes b, k, h and e. As one may observe, the source consumption along each subpath all does not exceed 5. For example, three resource consumptions between b and k are 5, 3 and 3, respectively. Whereas for subpath between h and e, these values are 5, 4 and 2, respectively.

When single resource constraint is considered ($r = 1$), the RMCPR reduces to the minimum cost path problem with relays (MCPPR) in the literature. Cabral (2005) first introduced the MCPPR in the context of a telecommunication network design problem, where relay points are necessary. Both RMCPR and MCPPR can be used to model many network

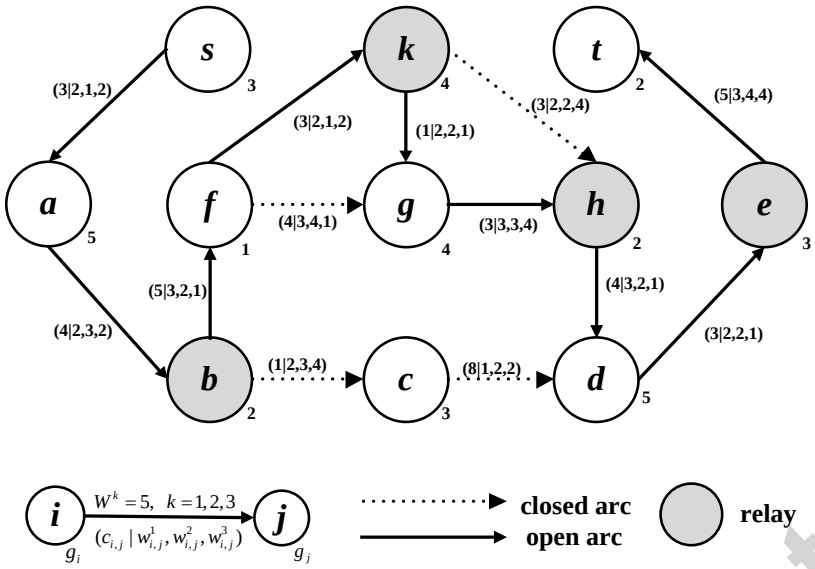


Figure 1 **Example of the RMCPR with Three Resources**

design problems in transportation and telecommunication systems (Cabral et al. 2008). Cabral et al. (2005) proposed three methods for the MCPPR, in which the most efficient has a complexity of $O(Wnm\log(W))$ (W is the upper bound on weights for locating relays). Laporte and Pascoal (2011) first modelled the MCPPR as a particular bicriteria path problem involving an aggregated function of the path and relay costs, as well as a weight function, and developed a more efficient labeling algorithm (LA) with a time complexity of $O(Wm + Wn\log(\max\{W, n\}))$. The LA is based on a labeling algorithm aided by an auxiliary matrix that stores labels with different weights. All the methods above for the MCPPR are pseudo-polynomial algorithms, which are strongly relying on instances in consideration. Note that the efficiency of the LA is closely related to the assumption that arc weights $w_{i,j}$ are integers. When arc weights are real, the number of possible labels will become extremely high, and thus incur a high computational cost. Additionally, one may argue that the LA can be directly applied to the RMCPR. We believe that it is impractical and inefficient because the number of possible labels will also become extremely high when there are several resources in consideration. Furthermore, the existence of multiple resources makes it very hard to verify the pruning criterion in the LA. We therefore need to develop a new and more efficient solution approach for the RMCPR.

Removal of relay setting requirement from the RMCPR results in the constrained shortest path problem (CSP) in the literature. The CSP consists of finding a path from a source

to a destination such that the total cost is minimized, subject to resource constraints (Lozano and Medaglia 2013). Solution methods for the CSP can be classified into three main categories: (1) dynamic programming (Dumitrescu and Boland 2003, Zhu and Wilhelm 2007, 2012), (2) path ranking methods (Handler and Zang 1980, Santos et al. 2007), and (3) Lagrangian relaxation (Carlyle et al. 2008, Formaneck and Cozzarin 2013). Methods based on dynamic programming are also known as label-setting or label-correcting algorithms. Zhu and Wilhelm (2007, 2012) developed a three-stage approach specialized for column generation subproblems that transforms the CSP into a shortest path problem and solves it using a labeling method. Handler and Zang (1980) solved the CSP by using a k-th shortest path algorithm; that is, they identify k paths, sort them by length, and evaluate them successively until they find the first path that satisfies the resource consumption constraint. Carlyle et al. (2008) proposed a method (CRW), which takes the general structure of the Lagrangian relaxation and enumeration approach with some subtle differences. Formaneck and Cozzarin (2013) proposed a modified Lagrangian relaxation method, which uses subgradient optimization instead of bisection search as in the CRW method. Aside from its straightforward application (i.e., shortest path subject to a time limit), the CSP and its variants naturally appears as a subproblem when more difficult problems are solved via column-generation approaches. The CSP is used as a column generation scheme for air cargo planning and routing (Derigs et al. 2009), crew pairing (Muter et al. 2013), tail assignment problem in aircraft scheduling (Gronkvist 2006), day-to-day crew operation (Stojkovic et al. 1998), and crew rostering problems (Gamache et al. 1999).

The RMCPR is also related to some network design problems with relays, such as the network design problem with relays (NDR) (Cabral et al. 2007), the directed network design problem with relays (DNDR) (Li et al. 2012), and the two-edge connected network design problem with relays (2ECON-NDPR)(Konak et al. 2009). In both NDR and DNDR, we need to construct routes for multiple commodities, and there is only one resource consumption ($r = 1$). Konak (2012) proposed a genetic algorithm based on special crossover and mutation for the NDR. For the DNDR, Li et al. (2012) gave node-arc and arc-path formulations, and developed a branch-and-price approach. The 2ECON-NDPR is an extension of the NDR by incorporating network survivability. Konak (2014) proposed a hybrid approach of a genetic algorithm and a Lagrangian heuristic for the 2ECON-NDPR. In the literature, researchers also studied combinatorial optimization problem with resource

constraints, e.g., the black and white traveling salesman problem (BWTSP) (Ghiani et al. 2006). The BWTSP is to design a shortest Hamiltonian cycle on a graph with black or white vertex, subject to upper bound constraints on the number of white vertices and on the length of the path between two consecutive black vertices (here number of white vertices and length can be viewed as two kinds of resources). Muter (2015) developed a column generation approach for the BWTSP.

Table 1 Characteristics of Typical Network Design Problems Related to the RMCPR

Problem	Features				
	relay requirement	♯resources	♯commodities	network	survivability requirement
CSP	no	multiple	single	directed	no
MCPPR	yes	single	single	directed	no
NDR	yes	single	multiple	undirected	no
DNDR	yes	single	multiple	directed	no
2ECON-NDPR	yes	single	multiple	undirected	yes
RMCPR	yes	multiple	single	directed	no

Table 1 summarizes characteristics of network design problems related to the RMCPR. As earlier mentioned, methods for problems in Table 1 cannot be directly applied to the RMCPR. We need to develop a more efficient solution approach.

Although the RMCPR is important in practice, it seems to have been largely ignored by the academic literature on network design. This paper aims to fill this gap. In this paper, we introduce and address the resource-constrained minimum cost path problem with relays. It is essentially a single-commodity multi-resource-constrained network design problem with relays. We present a pattern-chain formulation for the RMCPR. The pattern-chain formulation is a column generation formulation. We first introduce a concept of pattern, which is defined as a directed subpath linking two nodes such that only the ending point is relay point, and resource consumption constraints along this subpath are satisfied. We then transform the RMCPR into finding a chain of patterns (pattern chain), which links the source and destination, and present an integer column generation formulation to find the best pattern chain. We further prove that the relaxation of this integer column generation formulation can produce equivalently optimal solutions with variables being integers. As a result, we develop a column generation approach using the relaxed pattern-chain formulation, which is an exact algorithm for solving the RMCPR. At each iteration of this column generation approach, the pricing subproblem corresponding to a constrained shortest path

problem is solved to price out desirable columns (patterns) to enter the basis. The efficiency of our column generation approach depends on the effectiveness of the underlying pricing subproblem. To efficiently solve the pricing subproblem, we propose a Lagrangian relaxation based method. For the pricing subproblem associated with each node pair, this approach relaxes the resource consumption constraints, optimizes the resulting Lagrangian dual problem, and provides high-quality lower bounds on the reduced cost. The Lagrangian dual problem is solved by using subgradient optimization. Intermediate lower bounds may be used with different purposes: (1) terminate earlier solution of one pricing subproblem; (2) update the global lower bound on the reduced cost; or (3) reduce the size of set of node pairs considered in the pricing subproblem (refer to Section 3 for detailed implementation). If subgradient optimization cannot ensure no existence of desirable columns, then we need to further exactly solve the pricing subproblem with an mixed integer programming (MIP) formulation, using CPLEX. We evaluate our approach over 160 MCPPR instances (a special case of the RMCPR) with up to 1000 nodes and 200000 arcs, and 400 RMCPR instances with up to 1000 nodes, 300000 arcs and 30 resources. Using MCPPR instances, we first compare our approach with the best labeling algorithm, proposed by Laporte and Pascoal (2011). To demonstrate the performance of our approach on the RMCPR instances, we compare it with results returned by CPLEX with a node-arc formulation after one-hour implementation. Over MCPPR and RMCPR instances, computational results both show that our proposed approach is a computationally efficient exact method.

The remainder of this paper is organized as follows. In Section 2, we present a pattern-chain column generation formulation for the RMCPR. In Section 3, we present a column generation approach to exactly solve the RMCPR. In Section 4, we report computational experiments to evaluate our approach. We conclude this paper in Section 5.

2. Mathematical Formulation

In this section, we mainly present a pattern-chain formulation for the RMCPR, which is an integer column generation formulation. With this formulation, we will present an exact approach in Section 3. We first introduce a node-arc formulation for the RMCPR in the next subsection. Although it is not the main contribution in this paper, the node-arc formulation is used as a comparison benchmark to evaluate our exact approach proposed in Section 3. That is, the node-arc formulation is solved by MIP solver CPLEX to obtain upper bound of each instance in the computational experiments.

2.1. Node-arc Formulation

We first introduce a node-arc formulation, of which the underlying idea is similar to that for the DNDR in Li et al. (2012). Some binary variables and parameters are defined as follows:

- $x_{i,j}$: it equals 1 if arc (i,j) appears in the solution, and 0, otherwise;
- y_i: it equals 1 if node i is used as a relay node, and 0, otherwise;
- $v_{i,k}$: the total consumption of resource k after node i without visiting a relay. If node i is used as a relay in the path, $v_{i,k}$=0 for each k;
- b_i: it is equal to 1 if node i is the origin, -1 if node i is the destination, and 0, otherwise.

We present the node-arc formulation for the RMCPR as

$$\min f = \sum_{(i,j)\in A} c_{i,j}x_{i,j} + \sum_{i\in N} g_i y_i \tag{1}$$

$$\text{subject to } \sum_{(i,j)\in A} x_{i,j} - \sum_{(j,i)\in A} x_{j,i} = b_i \quad \forall i \in N, \tag{2}$$

$$v_{i,k} + w^k_{i,j}x_{i,j} - W^k(1 - x_{i,j} + y_j) \le v_{j,k} \quad \forall (i,j) \in A, \forall k \in K, \tag{3}$$

$$v_{i,k} + w^k_{i,j}x_{i,j} \le W^k \quad \forall (i,j) \in A, \forall k \in K, \tag{4}$$

$$0 \le v_{i,k} \le W^k(1 - y_i) \quad \forall i \in N, \forall k \in K, \tag{5}$$

$$v_{s,k} = 0 \quad \forall k \in K, \tag{6}$$

$$y_i \in \{0,1\} \quad \forall i \in N, \tag{7}$$

$$x_{i,j} \in \{0,1\} \quad \forall (i,j) \in A. \tag{8}$$

The objective function (1) minimizes the total network cost for setting arcs and relays. Constraints (2) are the mass balance constraints. Along each valid path, logical constraints for resource consumption are given in (3)-(5), and are justified as follows. If $x_{i,j} = 0$, then the set of constraints (3) become $v_{i,k} \le W^k(1 + y_j) + v_{j,k}$. They are redundant because any $v_{i,k} \le W^k$ and $v_{j,k} \ge 0$. If $x_{i,j} = 1$, then constraints (3) become $v_{i,k} + w^k_{i,j} - W^k y_j \le v_{j,k}$. Now if $y_j = 0$, then constraints (3) are valid. If $y_j = 1$, then constraints (3) are implied by constraints (4), which state that if arc (i,j) appears in the path, all the upper bound (W^k) cannot be violated. If node i is a relay node, then constraints (5) force $v_{i,k} = 0$. Constraints (6) state that for source node, $v_{s,k}$ should be zero with any resource k.

2.2. Pattern-chain Formulation

As one of our main contributions, we next present the pattern-chain formulation for the RMCPR. Before presenting the pattern-chain formulation, we first introduce concepts of pattern and pattern chain, respectively. A directed subpath from nodes i to j is referred to as a pattern, $p_{i,j}$, if the following two conditions are satisfied:

(1) For each resource k, the total resource consumption along this subpath does not exceed the given upper bound W^k, and

(2) Only the starting and ending nodes i and j are relay nodes.

We further assume that two relays are respectively located at source s and destination t, and they do not produce any cost, i.e., $g_s = g_t = 0$. Then, a feasible solution to the RMCPR can be viewed as a chain of patterns (pattern chain). For example, in Figure 1, one feasible solution corresponds to path $(s, a, b, f, k, g, h, d, e, t)$ with nodes b, k, h, and e being relays. Obviously, we have patterns $p_{s,b}$, $p_{b,k}$, $p_{k,h}$, $p_{h,e}$, and $p_{e,t}$. These patterns form a chain of patterns (pattern chain).

We refer to a node pair $< i, j >$ as "directly connected node pair (DCNP)", if there exists at least one pattern from i to j in network G. Let U be the set of all DCNPs. Given one DCNP $< i, j >$, there may exist different patterns between nodes i and j. For example, consider DCNP $< k, h >$ in Figure 1. Associated with this DCNP, there are two patterns: (k, h) with relay h, and (k, g, h) with relay h. Let $P_{i,j}$ be the set of all patterns associated with DCNP $< i, j >$. Let A_p be the set of arcs included in pattern p. Since the ending point of one pattern may be the starting point of another one, we exclude from calculation of the pattern cost the relay cost of the starting point. For each pattern $p \in P_{i,j}$, define $\delta^p_{i,j}$ as pattern cost, which is calculated as

$$\delta^p_{i,j} = \sum_{(u,v)\in A_p} c_{u,v} + g_j.$$

Following the above definition, the pattern cost consists of two parts: arc cost and relay cost. A pattern p with cost of $\min\{\delta^{p'}_{i,j} : p' \in P_{i,j}\}$ is called the minimal-cost pattern associated with DCNP $< i, j >$.

We refer σ to as a pattern chain if it defines a feasible solution to the RMCPR. Let $\delta(p)$ denote the cost of one pattern p, which is included in pattern chain σ (e.g., $p \in \sigma$). We further define the cost $\delta(\sigma)$ of pattern chain σ as

$$\delta(\sigma) = \sum_{p\in\sigma} \delta(p).$$

We now restate the RMCPR as follows:

DEFINITION 1. Given graph G and pattern set $P_{i,j}$ for each DCNP $< i, j > \in U$, find a pattern chain σ linking source s and destination t such that the cost of pattern chain, $\delta(\sigma)$ is minimized.

Further define binary variable $z_{i,j}^p$: it equals 1 if pattern p associated with node pair $< i, j >$ appears in the solution, and 0 otherwise. We next present a pattern-chain formulation (PC1) for the RMCPR:

$$(\mathbf{PC1}) \ \min f = \sum_{<i,j>\in U} \sum_{p\in P_{i,j}} \delta_{i,j}^p z_{i,j}^p \tag{9}$$

$$\text{subject to} \sum_{<i,j>\in U} \sum_{p\in P_{i,j}} z_{i,j}^p - \sum_{<j,i>\in U} \sum_{p\in P_{j,i}} z_{j,i}^p = b_i \quad \forall i \in N, \tag{10}$$

$$z_{i,j}^p \in \{0,1\} \quad \forall p \in P_{i,j}, \forall < i, j > \in U, \tag{11}$$

where objective function (9) minimizes the total cost of patterns. Constraints (10) are the flow balance constraints at each node. Note that PC1 is a formulation that results from Dantzig-Wolfe decomposition to the node-arc formulation (Dantzig and Wolfe 1960).

By relaxing variables $z_{i,j}^p$ to be linear, we next give formulation (PC2), and prove that this formulation must have an optimal solution with all integral $z_{i,j}^p$.

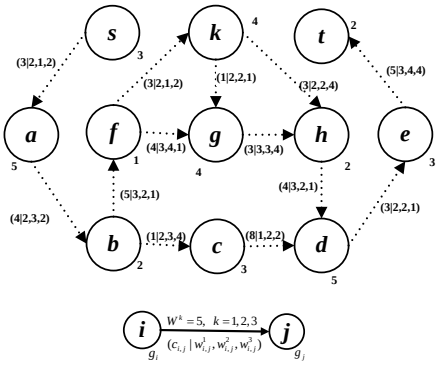
$$(\mathbf{PC2}) \ \min f = \sum_{<i,j>\in U} \sum_{p\in P_{i,j}} \delta_{i,j}^p z_{i,j}^p \tag{12}$$

$$\text{subject to} \sum_{<i,j>\in U} \sum_{p\in P_{i,j}} z_{i,j}^p - \sum_{<j,i>\in U} \sum_{p\in P_{j,i}} z_{j,i}^p = b_i \quad \forall i \in N, \tag{13}$$

$$0 \le z_{i,j}^p \le 1, \forall p \in P_{i,j}, \forall < i, j > \in U. \tag{14}$$

LEMMA 1. *There exists one optimal solution to formulation (PC2) with all $z_{i,j}^p$ as integers.*

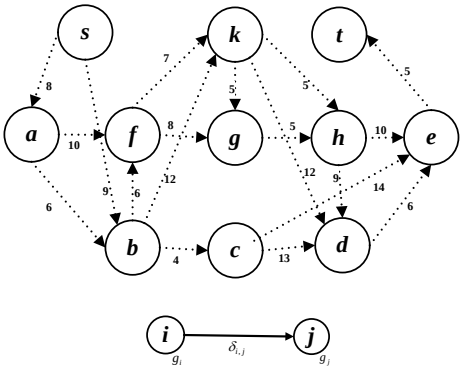
Proof. We prove this result using property for integer programming with totally unimodular matrix (Wolsey 1998). The constraint matrix is of the form $\begin{pmatrix} \Gamma \\ I \end{pmatrix}$ where Γ comes from constraints (13), and I is from the upper bound constraints. Following sufficient conditions for total unimodularity, it suffices to show that Γ is totally unimodular. Since b_i is 0 or ± 1 in (13), therefore formulation (PC2) has an optimal integral solution. This completes our proof. □



(a) Graph G

No.	Node pair	Node sequence	Pattern cost
1	<s, a>	(s, a)	8
2	<s, b>	(s, a, b)	9
3	<a, b>	(a, b)	6
4	<a, f>	(a, b, f)	10
5	<b, c>	(b, c)	4
6	<b, f>	(b, f)	6
7	<b, k>	(b, f, k)	12
8	<c, d>	(c, d)	13
9	<c, e>	(c, d, e)	14
10	<d, e>	(d, e)	6
11	<e, t>	(e, t)	5
12	<f, g>	(f, g)	8
13	<f, k>	(f, k)	7
14	<g, h>	(g, h)	5
15	<h, d>	(h, d)	9
16	<h, e>	(h, d, e)	10
17	<k, d>	(k, h, d)	12
18	<k, g>	(k, g)	5
19	<k, h>	(k, h)	5

(b) Pattern list



(c) Transformed graph G'

Figure 2 **Example of Network Transformation of the RMCPR**

LEMMA 2. *Each pattern included in one optimal solution is the minimal-cost pattern.*

Proof. The proof is straightforward from the definitions of pattern and pattern chain. □

Lemma 2 shows that the RMCPR reduces to a shortest path problem if we can efficiently find a minimum cost pattern for each DCNP. To illustrate this, consider a network transformation shown in Figure 2. Figure 2(a) shows the original network with three resources. Figure 2(b) lists all minimum-cost patterns subject to resource consumption constraints. Figure 2(c) gives the transformed graph G', in which each arc represents one pattern, and arc weight is equal to pattern costs listed in Figure 2(b). In Figure 2(c), the shortest path from nodes s to t is (s, b, c, e, t), which defines pattern chain $(p_{s,b}, p_{b,c}, p_{c,e}, p_{e,t})$ in Figure 2(a). That is, the optimal solution is path (s, a, b, c, d, e, t) with nodes b, c and e as relays.

Since finding the minimum cost pattern for a node pair corresponds to a CSP, the RMCPR is equivalent to all-pairs CSP plus a shortest path problem. Unfortunately, since the CSP is an $\mathcal{NP}$-hard problem, even for the case of single resource (Garey and Johnson 1979), it will require expensive computational time to enumerate all-pairs minimum cost patterns as the network scale tends to be large.

3. Column Generation Approach for the RMCPR

In this section, we propose a column generation approach (CG) using formulation (PC2). Following Lemma 1, the proposed CG is an exact approach for the RMCPR. Formulation (PC2) is a large formulation and has an enormous number of columns (variables) with one

variable for each valid pattern. It is impractical to explicitly enumerate all variables and then solve the formulation by MIP solvers. The underlying idea of the CG is not to explicitly list all of the columns of formulation (PC2), but rather to generate them only as needed. To be more precise, our CG starts with a "simple" formulation including a few columns, and prices out potential columns to enter the basis by solving the pricing subproblem. We design a Lagrangian relaxation based method to efficiently solve the pricing subproblem. If no column can be further priced out, then the RMCPR is solved to optimality. For detailed implementations of general column generation, readers may refer to Barnhart et al. (1994), Vanderbeck and Wolsey (1996), and Desaulniers et al. (2005). In addition, a CG based lower bound procedure for the NDR, and a branch-and-price approach for the DNDR, are respectively presented by Cabral et al. (2007) and Li et al. (2012).

3.1. Restricted Master Problem

The master problem of the RMCPR is formulated as PC2. In fact, only a very small subset of all columns will appear in an optimal solution, and other columns with non-negative reduced cost can be ignored. We refer to a master problem with only a subset of columns as the restricted master problem (RMP). In the context of the RMCPR, the RMP is defined by subsets $U' \subseteq U$ and $P'_{i,j} \subseteq P_{i,j}, \forall < i,j > \in U'$ as follows:

$$(\mathbf{RMP})\ \min f = \sum_{<i,j>\in U'} \sum_{p\in P'_{i,j}} \delta^p_{i,j} z^p_{i,j} \tag{15}$$

$$\text{subject to} \sum_{<i,j>\in U'} \sum_{p\in P'_{i,j}} z^p_{i,j} - \sum_{<j,i>\in U'} \sum_{p\in P'_{j,i}} z^p_{j,i} = b_i, \forall i \in N, \tag{16}$$

$$0 \le z^p_{i,j} \le 1, \forall p \in P'_{i,j}, \forall < i,j > \in U'. \tag{17}$$

After obtaining the LP solution to a RMP, we then check whether there exists some column with negative reduced cost, which is not included in the RMP. If one or more such columns do exist, we add all of them to the RMP and resolve the RMP. If none is found, the RMP is optimally solved and the current LP solution also defines an optimal solution to formulation (PC1).

3.2. Pricing Subproblem

For each RMP solution, the pricing subproblem is solved to identify whether there exists one column with negative reduced cost. Let π^i represent dual variable associated with each constraint in (13). In the pattern-chain formulation of the RMCPR, each column defines

a pattern for some node pair. From formulation (PC2), we can get the reduced cost $\overline{C}^p_{i,j}$ of one column associated with node pair $< i, j >$ as

$$\overline{C}^p_{i,j} = \delta^p_{i,j} - \pi^i + \pi^j \quad \forall p \in P_{i,j}, \forall < i, j > \in U. \tag{18}$$

To price out one column, we can solve one CSP associated with each node pair. For any node pair $< i, j > \in U$, let $\delta^*_{i,j}$ be the cost of the minimum-cost pattern p. Then, the master problem is optimally solved, if for each node pair $< i, j > \in U$, $\delta^*_{i,j} - \pi^i + \pi^j \geq 0$. Otherwise, desirable column is identified for node pair $< i, j >$, and the corresponding pattern p can be added to the RMP.

In the following, we present a Lagrangian relaxation based method for efficiently solving the pricing subproblem.

3.2.1. Formulation for the Pricing Subproblem

We first develop a mixed integer programming formulation for the pricing subproblem. For each node pair $< i, j >$, the pricing subproblem corresponds to a CSP, which consists of finding a minimum cost path from origin i to destination j, subject to the resource consumption constraints. Associated with node pair $< i, j >$, the MIP formulation for the pricing subproblem is as

$$(\mathbf{CSP}_{i,j}) \quad \delta^*_{i,j} = \min \sum_{(u,v)\in A} c_{u,v} x_{u,v} + g_j$$

$$\text{subject to} \sum_{(u,v)\in A} x_{u,v} - \sum_{(v,u)\in A} x_{v,u} = \begin{cases} 1, & \text{if } u = i, \\ -1, & \text{if } u = j, \\ 0, & \text{otherwise,} \end{cases} \qquad \forall u \in N, \tag{19}$$

$$\sum_{(u,v)\in A} x_{u,v} w^k_{u,v} \leq W^k, \forall k \in K, \tag{20}$$

$$x_{u,v} \in \{0, 1\}, \forall (u, v) \in A. \tag{21}$$

With this formulation, we may implement any MIP solver (e.g., CPLEX) to exactly solve the pricing subproblem. This method is effective only for small RMCPR instances, because the CSP is a well known $\mathcal{NP}$-hard problem. For large RMCPR instances, the number of node pairs will be very large, which thus requires expensive computational time to solve the pricing subproblem. Alternatively, we next introduce a Lagrangian relaxation based method, which is able to solve the pricing subproblem more efficiently.

3.2.2. Lagrangian Relaxation based Method for the Pricing Subproblem

Lagrangian relaxation is one of the major solution approaches for the CSP (Carlyle et al. 2008, Formaneck and Cozzarin 2013). We here discuss a Lagrangian relaxation based method to efficiently solve the pricing subproblem.

Given non-negative Lagrange multipliers $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \cdots, \lambda_r)$, we relax the resource consumption constraints (20) by bringing them into the objective function, and have the Lagrangian problem

$$\underline{\delta}_{i,j}(\boldsymbol{\lambda}) = \min \sum_{(u,v)\in A} c_{u,v}x_{u,v} - \sum_{k\in K}\lambda_k(W^k - \sum_{(u,v)\in A} x_{u,v}w^k_{u,v}) + g_j$$

$$\text{subject to (19) and (21).}$$

It is well known that $\delta^*_{i,j} \geq \underline{\delta}_{i,j}(\boldsymbol{\lambda})$. The Lagrangian lower bound $\underline{\delta}_{i,j}(\boldsymbol{\lambda})$ is then optimized through solving the following Lagrangian dual problem (LDP):

$$(\mathbf{LDP}_{i,j}) \quad \underline{\delta}^*_{i,j}(\boldsymbol{\lambda}) = \max_{\boldsymbol{\lambda}\geq \mathbf{0}} \underline{\delta}_{i,j}(\boldsymbol{\lambda}) = \max_{\boldsymbol{\lambda}\geq \mathbf{0}} \min \sum_{(u,v)\in A} (c_{u,v} + \sum_{k\in K}\lambda_k w^k_{u,v})x_{u,v} - \sum_{k\in K}\lambda_k W^k + g_j$$

$$\text{subject to (19) and (21).}$$

For any fixed $\boldsymbol{\lambda} \geq \mathbf{0}$, computing $\underline{\delta}_{i,j}(\boldsymbol{\lambda})$ simply requires solution of a shortest path problem with Lagrangian-modified arc weights.

Associated with pattern $p \in P_{i,j}$, the reduced cost is non-negative, if $\underline{\delta}_{i,j}(\boldsymbol{\lambda}) - \pi^i + \pi^j \geq 0$, for some $\boldsymbol{\lambda}$. Therefore, a high-quality lower bound $\underline{\delta}_{i,j}(\boldsymbol{\lambda})$ may terminate earlier solution of the pricing subproblem associated with one node pair. This is verified by our computational results.

In this paper, the LDP is solved by subgradient optimization (Ahuja et al. 1993). At each iteration τ of subgradient optimization, the Lagrange multipliers, $\boldsymbol{\lambda}$, are updated according to

$$\lambda^{(\tau+1)}_k = \lambda^{(\tau)}_k + h^{(\tau)}\xi^{(\tau)}_k, \tag{22}$$

where $\boldsymbol{\xi}^{(\tau)}$ is the direction of a subgradient, which is obtained as

$$\xi^{(\tau)}_k = \sum_{(u,v)\in A} w^k_{u,v}x^{(\tau)}_{u,v} - W^k, \forall k \in K, \tag{23}$$

where $x_{u,v}^{(\tau)}$ is the solution obtained by LDP at iteration τ. Step size $h^{(\tau)}$ specifies how far we move in the gradient direction. We heuristically select the step length as

$$h^{(\tau)} = \gamma^{(\tau)} \frac{\overline{\delta}_{i,j} - \underline{\delta}_{i,j}(\lambda^{(\tau)})}{\|\boldsymbol{\xi}^{(\tau)}\|^2}, \tag{24}$$

where $\overline{\delta}_{i,j}$ is the upper bound on the pattern cost, and $\overline{\delta}_{i,j} \geq \delta^*_{i,j}$. We first set $\overline{\delta}_{i,j}$ to a big value and update it, if we find a better pattern, as the algorithm proceeds. $\gamma^{(\tau)}$ was set to 0.6 in our implementation.

For each node pair $< i, j >$, we terminate the subgradient optimization, if one of the following three conditions is satisfied:

(1) Intermediate $\underline{\delta}_{i,j}(\boldsymbol{\lambda}) - \pi^i + \pi^j$ is non-negative: impossible to identify one desirable column.

(2) Iterations reach a certain maximum number τ_{max} (e.g., 200 was used in our implementation).

(3) Intermediate $\overline{\delta}_{i,j}$ is equal to $\underline{\delta}_{i,j}$: the Lagrangian dual optimally solves the pricing subproblem.

Note that the above implementation of the sugradient procedure cannot guarantee convergence. If the subgradient optimization reaches maximum iterations τ_{max} and conditions (1) and (3) are not satisfied, we then exactly solve the pricing subproblem, using MIP solvers, and verify if there exist desirable columns.

3.3. Algorithm Implementation

We next summarize and detail the implementation of the proposed solution approach CG. The overall structure of the CG is shown in Algorithm 1. Starting from a subset of columns (patterns) that obtained from node pair set U, our approach initializes the RMP and solved it optimally. At each iteration, using the method based on Lagrangian relaxation, the pricing subproblem is solved to price out a new column, which is then used to update the RMP, if it exists. The iteration continues until no new column can be identified. We now detail the implementation of the CG as follows.

3.3.1. Initialization of Node Pairs

Function InitializeNodePair(Q) is to initialize the set of directly connected node pairs (Algorithm 2). Let $A(i) = \{(i,j) \in A : j \in N\}$. Given arc weights $d_{i,j}$, Algorithm 3 returns

Algorithm 1. CG for the RMCPR

input instance data Q
U=InitializeNodePair(Q) (see Algorithm 2)
InitializeRMP(Q, U) (see Algorithm 4)
set *global_add* = true
while (*global_add* = true) **do**
  solve the RMP and get dual variables $\boldsymbol{\pi}$
  set *global_add* = false
  IdentifyDesirableColumns(Q, U_g, $\underline{\boldsymbol{\delta}}$, $\boldsymbol{\pi}$) (see Algorithm 5)
end while
output optimal solution

Algorithm 2. InitializeNodePair(Q)

set $U = \{< i, j > | i, j \in N \text{ and } i \neq j\}$
for each resource $k \in K$ **do**
  for each $< i, j > \in U$ **do**
    p = ShortestPath($Q, i, j, w^k_{u,v}$) (see Algorithm 3)
    if $w^k(p) > W^k$, **then** set $U = U \setminus \{< i, j >\}$ **end if**.
  end for
end for
return U

the shortest path from a source s_0 to a destination t_0. We here assume path p and pattern p are exchangeable, unless otherwise specified.

We initialize set U to include all possible node pairs in the network. Given a path p from nodes i to j in graph G, let $w^k(p)$ be the total consumption of resource k along arcs included in this path. For each node pair $< i, j >$ and each resource k, we then solve a shortest path problem with arc weights being resource consumption $w^k_{u,v}$. Cost $w^k(p)$ of the returned path p defines the lower bound of the consumption of resource k. If for at least one resource k, $w^k(p)$ exceeds the limit W^k, we can exclude $< i, j >$ from set U and thus prune it from the pricing subproblem. By this procedure, we may eliminate many "non-connected node pairs", and thus speed up solution of the pricing subproblem. Note that although InitializeNodePair(Q) cannot ensure the returned node pairs are all DCNPs, it can exclude some non-connected node pairs and thus reduce the size of the pricing subproblem at each iteration of the CG.

3.3.2. Initialization of the RMP

Function InitializeRMP(Q, U) is to determine a set of initial columns (patterns), which are used to initialize the starting RMP. For each node pair $< i, j >$, we solve a shortest path problem with arc weight being $c_{i,j}$. Let $C(p)$ be the corresponding path cost. Let $\underline{\delta}_{i,j}$ be the lower bound on the cost of patterns associated with node pair $< i, j >$, which is also

Algorithm 3. ShortestPath(Q, s_0, t_0, $d_{i,j}$)

set $S = \emptyset$ and $\overline{S} = N$
set $\mu_i = \infty, \forall i \in N$
set $\mu_{s_0} = 0$ and $pred(s_0) = 0$
while $t_0 \in \overline{S}$ **do**
  select $i \in \overline{S}$ with minimum μ_i
  set $S = S \cup \{i\}$ and $\overline{S} = \overline{S} \backslash \{i\}$
  for each arc $(i, j) \in A(i)$ **do**
    if $\mu_j > \mu_i + d_{i,j}$, **then** set $\mu_j = \mu_i + d_{i,j}$ and $pred(j) = i$
  end for
end while
retrieve and return shortest path p

Algorithm 4. InitializeRMP(Q, U)

set $U_g = U$
for each node pair $< i, j > \in U$ **do**
  set $\overline{\delta}_{i,j} = \infty$, $\underline{\delta}_{i,j} = 0$
  $p =$ ShortestPath$(Q, i, j, c_{u,v})$
  $\underline{\delta}_{i,j} = C(p) + g_j$
  for $k = 1$ to r **do**
    if $w^k(p) > W^k$, **then** break **end if**
    set $k = k + 1$
  end for
  if $k = r + 1$, **then**
    set $\overline{\delta}_{i,j} = \underline{\delta}_{i,j}$, $U_g = U_g \setminus \{< i, j >\}$, and add pattern to the RMP
  end if
end for
return RMP, $\underline{\boldsymbol{\delta}}$ and U_g

a lower bound on the objective value of CSP$_{i,j}$. Note that we have $\underline{\delta}_{i,j} = C(p) + g_j$. Two cases may happen:

(1) If the returned path p defines a pattern, i.e, satisfying the resource constraints, then this pattern should be minimal-cost pattern associated with node pair $< i, j >$, i.e, $\overline{\delta}_{i,j} = \underline{\delta}_{i,j}$ (e.g., this corresponds to LDP$_{i,j}$ with $\boldsymbol{\lambda} = \mathbf{0}$). We then add this pattern to the RMP. As a result, we can exclude node pair $< i, j >$ for the pricing subproblem.

(2) If path p does not define a pattern, we obviously obtain an initial lower bound $\underline{\delta}_{i,j}$ of CSP$_{i,j}$.

Let $\underline{\boldsymbol{\delta}} = \{\underline{\delta}_{i,j} :< i, j > \in U, i, j \in N\}$. Algorithm 4 also returns initial set U_g of node pairs in the pricing subproblem. Note that both $\underline{\boldsymbol{\delta}}$ and U_g will be updated during subgradient optimization (see Algorithm 5).

3.3.3. Solution of the Pricing Subproblem

We implement the proposed Lagrangian relaxation based method to identify desirable columns (see Algorithm 5). Let M be a very big value (e.g., 10000 in our implementation). For each node pair, the Lagrangian relaxation based method works as follows:

Algorithm 5. IdentifyDesiraableColumns(Q, U_g, $\underline{\boldsymbol{\delta}}$, $\boldsymbol{\pi}$)

for each node pair $<i,j>\in U_g$
  if $\underline{\delta}_{i,j}-\pi^i+\pi^j<0$, **then**
    set $\overline{\delta}_{i,j}=M$, $\tau=0$, $h^{(0)}=0$, and $\lambda_k^{(0)}=0,\forall k\in K$
    while $\tau<\tau_{max}$ **do**
      compute $Lr_{u,v}^{(\tau)},\forall(u,v)\in A$
      $p=$ ShortestPath$(Q,i,j,Lr_{u,v}^{(\tau)})$
      compute $\underline{\delta}_{i,j}(\boldsymbol{\lambda})$
      if $\underline{\delta}_{i,j}<\underline{\delta}_{i,j}(\boldsymbol{\lambda})$, **then** set $\underline{\delta}_{i,j}=\underline{\delta}_{i,j}(\boldsymbol{\lambda})$ **end if**
      if $w^k(p)\le W^k$, $\forall k\in K$, **then**
        if $\overline{\delta}_{i,j}>C(p)+g_j$, **then**
          set $\overline{\delta}_{i,j}=C(p)+g_j$
          if $\overline{\delta}_{i,j}-\underline{\delta}_{i,j}<\epsilon$, **then**
            set $U_g=U_g\setminus\{<i,j>\}$
            break
          end if
        end if
      end if
      if $\underline{\delta}_{i,j}-\pi^i+\pi^j\ge 0$, **then** break **end if**
      update $\lambda_k^{(\tau)}$ according to (22)-(24)
      $\tau=\tau+1$
    end while
    if $\tau=\tau_{max}$, **then**
      solve $\text{CSP}_{i,j}$ using CPLEX with a starting solution associated with $\overline{\delta}_{i,j}$
      set $\overline{\delta}_{i,j}=\delta_{i,j}^*$, $\underline{\delta}_{i,j}=\delta_{i,j}^*$
    end if
    if $\underline{\delta}_{i,j}-\pi^i+\pi^j<0$, **then**
      set *global_add*=true
      add identified column to the RMP
      set $U_g=U_g\setminus\{<i,j>\}$
    end if
  end if
end for

We first implement a speed-up strategy. That is, if $\underline{\delta}_{i,j}-\pi^i+\pi^j\ge 0$, then for each pattern $p\in P_{i,j}$, the associated reduced cost is non-negative. This node pair is excluded from the pricing subproblem.

We then solve the Lagrangian dual problem $\text{LDP}_{i,j}$. At each iteration τ of the subgradient optimization, we solve one shortest path problem associated with each node pair $<i,j>$, in which arc weight is calculated as

$$Lr_{u,v}^{(\tau)}=c_{u,v}+\sum_{k\in K}\lambda_k^{(\tau)}w_{u,v}^k,\forall(u,v)\in A.$$

Let the $Lr^{(\tau)}(p)$ be the cost of the returned shortest path p. Thus the corresponding current lower bound $\underline{\delta}_{i,j}(\boldsymbol{\lambda})$ is calculated as

$$\underline{\delta}_{i,j}(\boldsymbol{\lambda})=Lr^{(\tau)}(p)-\sum_{k\in K}\lambda_k^{(\tau)}W^k+g_j,$$

which is used to update the best lower bound $\underline{\delta}_{i,j}$.

At each iteration τ of subgradient optimization, one of the three cases may happen:

(1) $\underline{\delta}_{i,j} - \pi^i + \pi^j \geq 0$: no column can be priced out for node pair $< i, j >$ in subsequent iterations, and thus it is able to terminate solution of $\text{LDP}_{i,j}$ earlier.

(2) The returned shortest path p defines a pattern (e.g., resource consumption constraints are satisfied): the upper bound $\overline{\delta}_{i,j}$ may be updated.

(3) $\overline{\delta}_{i,j} - \underline{\delta}_{i,j} < \epsilon$ (ϵ is a very small value, e.g., 0.0001): the $\text{CSP}_{i,j}$ is optimally solved. If objective coefficients are integers, we can set ϵ to 1.

After solution of each LDP terminates, the best lower bound $\underline{\delta}_{i,j}$ is used to check if there exists desirable column associated with node pair $< i, j >$. We may need to further solve $\text{CSP}_{i,j}$ using CPLEX with a starting feasible solution found in the subgradient optimization, and identify possible column.

In our implementation, the CG approach is terminated when each instance is optimally solved. That is, no column with negative reduced cost can be priced out.

4. Computational Evaluation

We next carry out a series of experiments to evaluate our proposed algorithm. The CG was coded in C language and compiled on VC++ 2012. The RMP and CSP associated with the pricing subproblem were solved by CPLEX 12.5. Experiments were conducted on a 2.5GHz PC with 8Gb RAM. Computing times are reported in seconds. The purposes of our experiments include:

- How well does the proposed approach perform on randomly generated MCPPR instances, which are special cases of the RMCPR (experiment 1)?
- How well does the proposed approach perform on randomly generated RMCPR instances (experiment 2)?

4.1. Test Instances

4.1.1. MCPPR Instances

As previously stated, the MCPPR is a special case of the RMCPR. Therefore, our CG approach can be used to solve the MCPPR. Laporte and Pascoal (2011) tested their LA on random MCPPR instances with up to 10000 nodes and 100000 arcs. We kindly appreciate that Prof. Pascoal shared their code with us. Both LA and our method were tested on our randomly generated instances (Set 1). The original implementation of LA assumes

that each arc weight $w_{i,j}$ is integer. To make LA properly run on instances with rational weights, we introduce a multiplier ϕ, which is used to amplify real weights $w'_{i,j}$ and W' to be integers. For example, instances with $W' = 100$ and three-decimal weight $w'_{i,j}$ are equivalent to instances with $W = 1000 \times W' = 100000$ and $w_{i,j} = w'_{i,j} \times 1000$, for a given $\phi = 1000$.

In total, we generated 16 groups of 160 MCPPR instances, with 10 instances in each group. Each instance is characterized by three parameters: n (the number of nodes), $\chi \in [0, 1]$ (the percentage of arcs emanating from a node), and ϕ. We generated instances as follows. To ensure instance feasibility, we first randomly constructed a directed Hamiltonian route including n nodes with starting and ending points being same, and initialized the network. Given network density χ, we then randomly generated $\lceil \chi n(n-1) - n \rceil$ arcs and added them to the network. Clearly, the larger n and χ are, the more arcs are in the network. Arc cost $c_{i,j}$ is integer uniformly generated from interval $[10, 70]$. Relay cost g_i is an integer randomly generated from interval $[20, 50]$. We set different magnitudes of ϕ. Instances with larger ϕ is closer to the case that weight value is real. W is set to $\phi \times W'$ with $W' = 100$. $w_{i,j}$ are integers generated from interval $[\frac{W}{10}, \frac{W}{2}]$. We randomly chose origin s and destination t in the network.

Table 2 summarizes characteristics of MCPPR instances in Set 1.

4.1.2. RMCPR instances

In experiment 2, we aim to evaluate the performance of the CG on RMCPR instances. Each RMCPR instance is characterized by three parameters: n, $\chi \in [0, 1]$ and r. To generate the network, we used a method which is similar to the one shown in Section 4.1.1. We randomly chose origin s and destination t. Arc costs $c_{i,j}$ are integers uniformly generated from interval $[10, 50]$. Relay costs g_i are integers randomly generated from interval $[20, 70]$. For each resource k, the resource consumption limit W^k is randomly chosen from $\{60, 80, 100, 120\}$, and weights $w^k_{i,j}$ are integers generated from interval $[10, 50]$.

Tables 3 and 4 respectively summarize characteristics of small RMCPR instances (Set 2) and large RMCPR instances (Set 3). In total, we generated 40 groups of RMCPR instances, with 10 instances in each group.

Table 2 Characteristics of MCPPR Instances in Set 1

Instance group	n	χ	ϕ	m
1	500	0.1	100	25000
2	500	0.1	1000	25000
3	500	0.1	2000	25000
4	500	0.1	4000	25000
5	500	0.2	100	50000
6	500	0.2	1000	50000
7	500	0.2	2000	50000
8	500	0.2	4000	50000
9	1000	0.1	100	100000
10	1000	0.1	1000	100000
11	1000	0.1	2000	100000
12	1000	0.1	4000	100000
13	1000	0.2	100	200000
14	1000	0.2	1000	200000
15	1000	0.2	2000	200000
16	1000	0.2	4000	200000

Table 3 Characteristics of Small RMCPR Instances in Set 2

Instance group	n	χ	r	m
1	200	0.1	5	4000
2	200	0.1	10	4000
3	200	0.1	15	4000
4	200	0.1	20	4000
5	200	0.2	5	8000
6	200	0.2	10	8000
7	200	0.2	15	8000
8	200	0.2	20	8000
9	200	0.3	5	12000
10	200	0.3	10	12000
11	200	0.3	15	12000
12	200	0.3	20	12000
13	500	0.1	5	25000
14	500	0.1	10	25000
15	500	0.1	15	25000
16	500	0.1	20	25000
17	500	0.2	5	50000
18	500	0.2	10	50000
19	500	0.2	15	50000
20	500	0.2	20	50000
21	500	0.3	5	75000
22	500	0.3	10	75000

4.2. Test Methods

To demonstrate the performance of the proposed approach, we attempt to compare it with other available methods for the MCPPR and RMCPR. The methods considered in our comparison experiments are shown in Table 5.

In experiment 1, we compare our approach with the labeling algorithm on randomly generated MCPPR instances in Set 1. Laporte and Pascoal (2011) developed four versions of LA: two versions of a label correcting algorithm (e.g., LC and LCP), and two similar

Table 4 Characteristics of Large RMCPR Instances in Set 3

Instance group	n	χ	r	m
23	500	0.3	15	75000
24	500	0.3	20	75000
25	1000	0.1	5	100000
26	1000	0.1	10	100000
27	1000	0.1	15	100000
28	1000	0.1	20	100000
29	1000	0.1	30	100000
30	1000	0.15	15	150000
31	1000	0.15	20	150000
32	1000	0.15	30	150000
33	1000	0.2	10	200000
34	1000	0.2	15	200000
35	1000	0.2	20	200000
36	1000	0.2	30	200000
37	1000	0.3	10	300000
38	1000	0.3	15	300000
39	1000	0.3	20	300000
40	1000	0.3	30	300000

Table 5 Methods in the Comparison Study

Experiment	Instance set	Methods in comparison
1	Set 1	CG, LCP
2	Sets 2 and 3	CG, CPLEX

variants of a label setting algorithm (e.g., LS and LSP). The latter two algorithms can be interrupted when a label associated with t is selected. Experimental results in Laporte and Pascoal (2011) show that the LCP is always the best one among four versions of LA. We thus compare our approach with the LCP in this paper. We ran the code of LCP on our computer and reported computational results.

In experiment 2, we want to demonstrate the ability of our approach in solving the RMCPR instances. Since there is no algorithm available in the literature, we ran CPLEX to solve each instance with a time limit of 3600 seconds. The integer solutions at termination are reported as comparison benchmark.

4.3. Comparison Results on MCPPR Instances

We first compare the CG with the LCP, using 160 randomly generated MCPPR instances. The computational results are reported in Table 6. Figure 3 displays the average running time of two algorithms under different levels of W. For each instance group, we introduce three performance measures:

- Obj: the average objective value of best solutions found by each algorithm;
- Avg.: the average CPU time required to solve instances in each group;
- #OPT: the number of instances optimally solved in each group.

Table 6 Comparison Results for MCPPR Instances (Set 1)

Instance group	LCP			CG		
	Obj	Avg.	#OPT	Obj	Avg.	#OPT
1	81.3	0.56	10	81.3	2.8	10
2	82.8	5.5	10	82.8	3.3	10
3	71.3	16.0	10	71.3	2.9	10
4	79.3	27.7	10	79.3	2.9	10
5	66.9	0.8	10	66.9	3.0	10
6	67.7	7.7	10	67.7	3.0	10
7	64.8	15.6	10	64.8	2.8	10
8	55.8	26.8	10	55.8	2.9	10
9	75.0	1.9	10	75.0	13.4	10
10	71.8	18.9	10	71.8	13.6	10
11	73.8	46.5	10	73.8	13.5	10
12	-	-	0	75.6	13.5	10
13	63.8	2.6	10	63.8	13.4	10
14	67.8	28.7	10	67.8	13.4	10
15	68.5	61.5	10	68.5	13.6	10
16	-	-	0	68.3	13.3	10

"-" indicates that LCP terminated with error "out of memory"

Computational results in Table 6 show that the CG solved all 160 instances to optimality. As one may observe, the LCP could not solve two groups of 20 instances with larger W. This is due to that greater W implies a larger label matrix for the LCP and thus leads to a higher number of labels, which makes the algorithm terminate with error "out of memory". Results in Figure 3 shows that for a fixed n and χ, the running time of the LCP significantly grows with the value of W. This demonstrates that the computational efficiency of LCP is significantly related to the upper bound W. However, the curve of computational times for the CG is flat. That is, the performance of the CG did not depend on the value of W.

4.4. Comparison Results on Small RMCPR Instances

We next implement the CG to solve 220 small RMCPR instances in Set 2. As a comparison benchmark, CPLEX was also run to solve these instances with the node-arc formulation. We set a time limit of 3600 seconds for two methods. Table 7 summarizes the computational results. In Table 7, Max and Min respectively denote the maximum and minimum CPU times required to solve instances in each group. Figure 4 shows the average computational time over instance group.

Computational results in Table 7 show that both CG and CPLEX could solve these 220 instances to optimality. Although two methods could deal with these instances, the CG significantly outperformed CPLEX in terms of average computing times required for solving instances. For example, the CG solved each instance within about 4 seconds. Whereas,

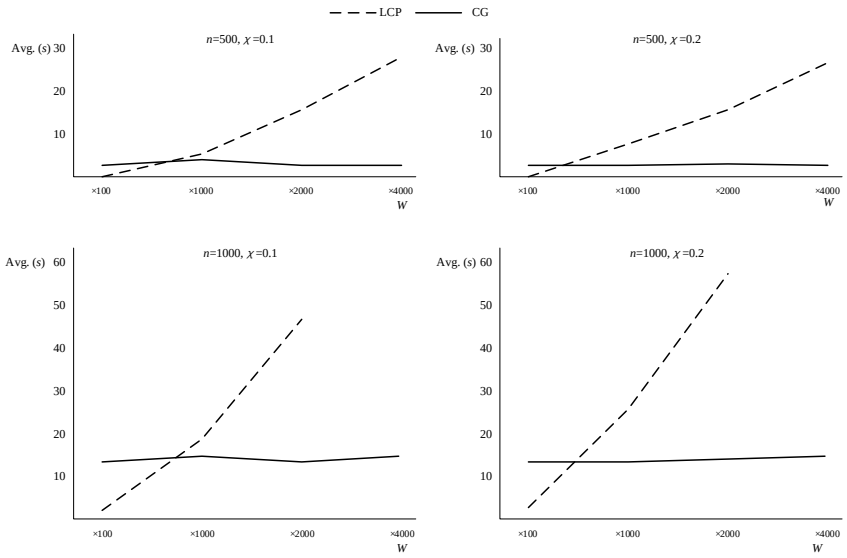


Figure 3 **Average Computational Times of Algorithms versus W**

Table 7 **Computational Results on Small RMCPR Instances (Set 2)**

Instance group	CPLEX					CG				
	Obj	Avg.	Max	Min	#OPT	Obj	Avg.	Max	Min	#OPT
1	102.9	2.2	3.7	1.6	10	102.9	0.9	2.7	0.2	10
2	113.2	4.6	5.7	3.4	10	113.2	1.1	6.7	0.2	10
3	128.8	7.6	10.8	4.6	10	128.8	1.5	4.8	0.1	10
4	95.6	7.4	11.9	4.6	10	95.6	0.5	3.7	0.2	10
5	89.9	4.1	6.0	2.8	10	89.9	1.1	3.4	0.2	10
6	92.0	7.8	14.5	4.5	10	92.0	2.3	3.3	0.2	10
7	86.5	13.3	18.3	7.1	10	86.5	1.1	1.6	0.2	10
8	94.9	15.7	23.1	9.1	10	94.9	1.5	4.7	0.2	10
9	77.2	6.0	7.7	4.8	10	77.2	0.9	1.7	0.4	10
10	71.7	11.2	18.2	7.1	10	71.7	1.1	2.7	0.3	10
11	82.9	17.3	22.6	9.5	10	82.9	1.9	3.2	0.3	10
12	85.1	25.1	30.3	14.9	10	85.1	2.0	2.7	0.3	10
13	92.7	13.4	21.4	8.9	10	92.7	3.0	4.0	1.6	10
14	108.3	29.3	37.4	24.2	10	108.3	6.7	9.3	1.0	10
15	88.1	46.2	76.3	22.9	10	88.1	6.3	8.1	1.0	10
16	91.7	59.6	83.3	25.8	10	91.7	6.9	7.7	0.9	10
17	71.7	20.7	27.6	17.2	10	71.7	4.2	7.0	2.0	10
18	83.0	71.7	96.6	50.5	10	83.0	8.5	18.0	1.7	10
19	79.6	100.0	124.2	38.7	10	79.6	11.8	19.4	1.6	10
20	78.2	120.0	167.6	48.6	10	78.2	12.0	21.5	1.2	10
21	60.8	31.3	50.1	27.0	10	60.8	2.5	6.7	1.3	10
22	70.4	60.1	95.0	44.0	10	70.4	7.8	35.6	1.3	10
Average		30.7					3.9			

CPLEX needed about 31 seconds for solving each instance. Results in Figure 4 show that computational efficiency of the CG fluctuated weakly over instance groups, in comparison with CPLEX.

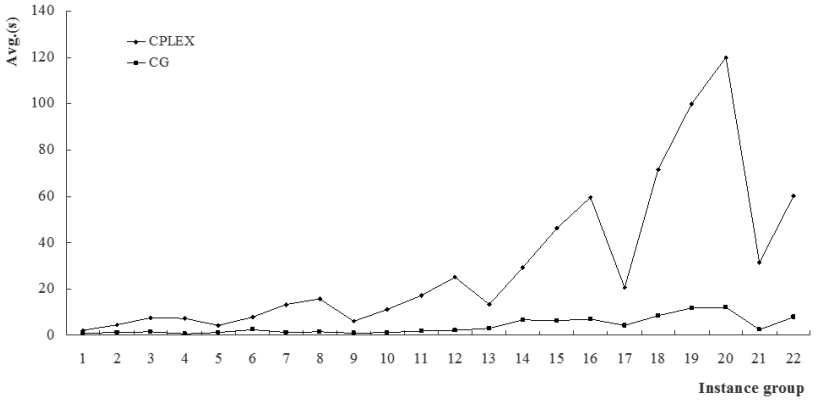


Figure 4 Average Computational Times Over Small RMCPR Instances (Set 2)

Table 8 Computational Results on Large RMCPR instances (Set 3)

Instance group	CPLEX					CG				
	Obj	Avg.	Max	Min	#OPT	Obj	Avg.	Max	Min	#OPT
23	85.5	326.1	413.9	285.0	10	85.5	23.4	38.3	1.9	10
24	79.1	356.7	405.9	296.1	10	79.1	24.3	42.9	1.9	10
25	78.8	49.3	66.1	40.0	10	78.8	16.1	34.7	6.4	10
26	90.8	199.5	241.5	164.4	10	90.8	29.7	43.2	5.3	10
27	103.8	301.0	365.7	135.9	10	103.8	55.4	135.4	6.0	10
28	98.7	405.3	1086.1	214.3	10	98.7	64.5	176.4	5.8	10
29	109.2	3313.4	3600	2750.2	4	99.4	41.3	92.5	6.0	10
30	82.6	2459.7	3600	1513.0	6	79.4	54.2	92.3	7.9	10
31	98.5	3600	3600	3600	0	92.2	84.5	101.7	68.3	10
32	-	3600	3600	3600	0	83.2	74.8	126.7	10.9	10
33	79.3	346.4	533.3	143.1	10	79.3	51.1	156.2	6.2	10
34	-	3600	3600	3600	0	71.7	61.4	191.3	5.8	10
35	-	3600	3600	3600	0	77.8	118.6	238.9	6.1	10
36	-	3600	3600	3600	0	83.8	239.2	389.1	149.7	10
37	-	3600	3600	3600	0	65.3	29.9	49.4	7.4	10
38	-	3600	3600	3600	0	76.2	146.9	675.7	7.3	10
39	-	3600	3600	3600	0	73.9	255.6	690.1	7.1	10
40	-	3600	3600	3600	0	72.4	510.9	949.5	78.2	10
Average		2231.0			4.4		104.5			10

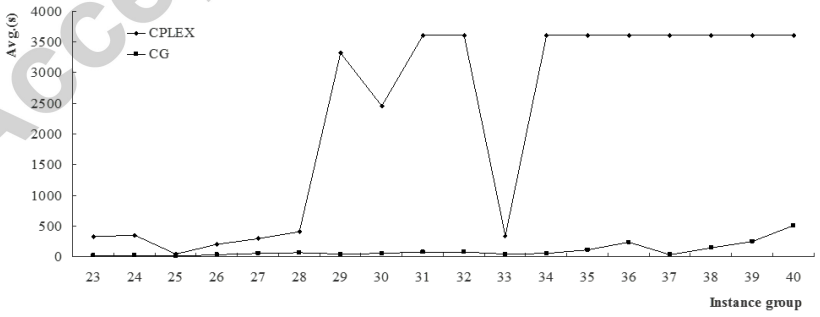


Figure 5 Average Computational Times Over Large RMCPR Instances (Set 3)

4.5. Comparison Results on Large RMCPR Instances

We finally evaluate the performance of our CG using 180 large RMCPR instances (Set 3). Again CPLEX was also run to solve each instance with a time limit of 3600 seconds. The computational results are summarized in Table 8. Entry "-" indicates that at termination, CPLEX did not find a feasible solution to any instance in each group. Figure 5 shows the average computational time over instance group.

As one may observe from the results in Table 8, instances in Set 3 are much harder than those in Set 2. Among these 180 instances, CPLEX solved only 80 instances within one hour. For eight groups of 80 instances, CPLEX even did not find any feasible solution at termination. The computational results clearly show that the CG worked significantly better than CPLEX. The CG solved all these 180 instances to optimality. For those 80 instances that CPLEX optimally solved, the CG found optimal solutions with significantly less computational time. That is, CPLEX required average 519.7 seconds to solve each instance, whereas the CG only needed 45 seconds.

In summary, our experimental results show that the CG is a computationally efficient approach for the MCPPR and RMCPR.

5. Conclusions

In this paper, we introduced and addressed the resource-constrained minimum cost path problem with relays, which arises in telecommunication and transportation systems. We first presented a pattern-chain formulation and then developed a column generation approach, which is able to optimally solve the RMCPR. In the column generation approach, we proposed a Lagrangian relaxation based method to efficiently solve the pricing subproblem, which aims at finding a desirable pattern associated with each node pair. The Lagrangian dual of each pricing subproblem was solved to check whether we can prune one node pair from the pricing subproblem and reduce its size. A series of experiments were carried out to evaluate our approach. With 160 MCPPR instances, we first compared our approach with the best labeling algorithm, of which the efficiency is closely related to the assumption that arc weights are integers. While finding solutions with same quality, our approach has a good performance in terms of computing time. Over 400 RMCPR instances, we finally compared our approach with CPLEX, which solved instances with the node-arc formulation with a time limit of one hour. The computational results show

that our approach outperformed CPLEX in terms of both solution quality and computing time, especially for large instances.

In the future work, we may extend our results to the resource-constrained multicommodity network design problem with relays. We plan to report on these developments in the near future.

Acknowledgments

This research has been partially supported by the Natural Science Foundation of China (Grant no. 71101105, 71532015), Program for New Century Excellent Talents in University (Grant no. NCET-13-0424), the Ministry of Education Humanities and Social Science Project (Grant no. 15YJA630030), research fund for doctoral program of higher education of China (Grant no. 20110072120013), Shanghai philosophical and social science program (Grant no. 2011EGL004), and the Fundamental Research Funds for the Central Universities. The research of the fourth author was partially supported by a discovery research grant from the Natural Science Engineering Research Council of Canada (Grant no. 87677).

References

Ahuja, R., T. Magnanti, J. Orlin. 1993. *Network flows: Theory, algorithms, and applications*. Prentice-Hall,Englewood Cliffs, NJ.

Barnhart, C., C. A. Hane, E. L. Johnson, G. Sigismondi. 1994. A column generation and partitioning approach for multi-commodity flow problems. *Telecommunication Systems* **3** 239–258.

Cabral, E. A. 2005. Wide area telecommunication network design: problems and solution algorithms with application to the alberta supernet. Ph.D. thesis, School of Business, University of Alberta.

Cabral, E. A., E. Erkut, G. Laporte. 2005. The shortest path problem with relays. *unpublished manuscript* .

Cabral, E. A., E. Erkut, G. Laporte, R. A. Patterson. 2007. The network design problem with relays. *European Journal of Operational Research* **180** 834–844.

Cabral, E. A., E. Erkut, G. Laporte, R. A. Patterson. 2008. Wide area telecommunication network design: application to the alberta supernet. *Journal of the Operational Research Society* **59** 1460–1470.

Carlyle, W. M., J. O. Royset, R. K. Wood. 2008. Lagrangian relaxation and enumeration for solving constrained shortest-path problems. *Networks* **52** 256–270.

Dantzig, G., P. Wolfe. 1960. Decomposition principle for linear programs. *Operations Research* **8** 101–11.

Derigs, U., S. Friederichs, S. Schafer. 2009. A new approach for air cargo network planning. *Transportation Science* **43** 370–380.

Desaulniers, Guy, Jacques Desrosiers, Marius M. Solomon. 2005. *Column Generation*. Springer US.

Dumitrescu, I., N. Boland. 2003. Improved preprocessing, labeling and scaling algorithms for the weight-constrained shortest path problem. *Networks* **42** 135–153.

Formaneck, S. D., B. P. Cozzarin. 2013. Technology adoption and training practices as a constrained shortest path problem. *Omega-International Journal of Management Science* **41** 459–472.

Gamache, M., F. Soumis, G. Marquis, J. Desrosiers. 1999. A column generation approach for large-scale aircrew rostering problems. *Operations Research* **47** 247–263.

Garey, M. R., D. S. Johnson. 1979. *Computers and intractability: a guide to the theory of NP-completeness*. New York: W.H. Freeman.

Ghiani, G., G. Laporte, F. Semet. 2006. The black and white traveling salesman problem. *Operations Research* **54** 366–378.

Gronkvist, M. 2006. Accelerating column generation for aircraft scheduling using constraint propagation. *Computers & Operations Research* **33** 2918–2934.

Handler, G. Y., I. Zang. 1980. A dual algorithm for the constrained shortest path problem. *Networks* **10** 293–310.

Konak, A. 2012. Network design problem with relays: A genetic algorithm with a path-based crossover and a set covering formulation. *European Journal of Operational Research* **218** 829–837.

Konak, A. 2014. Two-edge disjoint survivable network design problem with relays: a hybrid genetic algorithm and lagrangian heuristic approach. *Engineering Optimization* **46** 130–145.

Konak, A., S. Kulturel-Konak, A. E. Smith. 2009. Two-edge disjoint survivable network design problem with relays. *Operations Research And Cyber-Infrastructure* 279–292.

Laporte, G., M. M. B. Pascoal. 2011. Minimum cost path problems with relays. *Computers & Operations Research* **38** 165–173.

Li, X. Y., Y. P. Aneja, J. Z. Huo. 2012. Using branch-and-price approach to solve the directed network design problem with relays. *Omega-International Journal of Management Science* **40** 672–679.

Lozano, L., A. L. Medaglia. 2013. On an exact method for the constrained shortest path problem. *Computers & Operations Research* **40** 378–384.

Muter, I. 2015. A new formulation and approach for the black and white traveling salesman problem. *Computers & Operations Research* **53** 96–106.

Muter, I., S. I. Birbil, K. Bulbul, G. Sahin, H. Yenigun, D. Tas, D. Tuzun. 2013. Solving a robust airline crew pairing problem with column generation. *Computers & Operations Research* **40** 815–830.

Santos, L., J. Coutinho-Rodrigues, J. R. Current. 2007. An improved solution algorithm for the constrained shortest path problem. *Transportation Research Part B-Methodological* **41** 756–771.

Stojkovic, M., F. Soumis, J. Desrosiers. 1998. The operational airline crew scheduling problem. *Transportation Science* **32** 232–245.

Üster, H., P. Kewcharoenwong. 2011. Strategic design and analysis of a relay network in truckload transportation. *Transportation Science* **45** 505–523.

Vanderbeck, F., L. A. Wolsey. 1996. An exact algorithm for ip column generation. *Operations research letters* **19** 151–159.

Willoughby, K. A., D. H. Uyeno. 2001. Resolving splits in location/allocation modeling: a heuristic procedure for transit center decisions. *Transportation Research Part E-Logistics And Transportation Review* **37** 71–83.

Wolsey, L. A. 1998. *Integer Programming*. John Wiley and Sons, Inc., New York.

Zhu, X., W. E. Wilhelm. 2007. Three-stage approaches for optimizing some variations of the resource constrained shortest-path sub-problem in a column generation context. *European Journal of Operational Research* **183** 564–577.

Zhu, X., W. E. Wilhelm. 2012. Implementation of a three-stage approach for the dynamic resource-constrained shortest-path sub-problem in branch-and-price. *Computers & Operations Research* **40** 385–394.