Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2015, Article ID 803135, 7 pages
http://dx.doi.org/10.1155/2015/803135

Research Article

A Multiple-Starting-Path Approach to the Resource-Constrained kth Elementary Shortest Path Problem

Hyunchul Tae and Byung-In Kim

Department of Industrial and Management Engineering, Pohang University of Science and Technology (POSTECH), Pohang 790-784, Republic of Korea

Correspondence should be addressed to Byung-In Kim; bkim@postech.ac.kr

Received 12 September 2014; Accepted 9 March 2015

Academic Editor: Dong Ngoduy

The resource-constrained elementary shortest path problem (RCESPP) aims to determine the shortest elementary path from the origin to the sink that satisfies the resource constraints. The resource-constrained kth elementary shortest path problem (RCKESPP) is a generalization of the RCESPP that aims to determine the kth shortest path when a set of $k-1$ shortest paths is given. To the best of our knowledge, the RCKESPP has been solved most efficiently by using Lawler's algorithm. This paper proposes a new approach named multiple-starting-path (MSP) to the RCKESPP. The computational results indicate that the MSP approach outperforms Lawler's algorithm.

1. Introduction

The vehicle routing problem (VRP) is a well-known combinatorial problem of determining the optimal routes used by a fleet of vehicles to visit all vertices with the minimum cost. One of the most effective exact approaches for the VRP is branch-and-price (B&P). A B&P solves a linear relaxation of the set covering formulation of the VRP by means of column generation method at each node. The method solves the set covering relaxation by decomposing it to master and auxiliary problems. Whenever a master problem is solved, the dual values of its constraints are allocated to vertices as prizes. Then, an auxiliary problem is solved to find a column with a negative reduced cost.

In the VRP, the auxiliary problem exhibits a form of the resource-constrained elementary shortest path problem (RCESPP). The RCESPP aims to determine the shortest elementary path from the origin to the sink that satisfies the resource constraints. In the RCESPP, the cost of a path is calculated as the sum of the travel costs of the traversed arcs minus the sum of the prizes of the visited customers. Because of the prizes, the graph of the RCESPP may contain negative arcs and cycles. The RCESPP is strongly NP-Hard [1] and has been solved most efficiently by the dynamic programming (DP) algorithms [2–5].

Some VRP researches [6, 7] have opted to relax the elementary constraint of the RCESPP in their B&P because its relaxed version can be solved much faster. However, others [8, 9] opted not to because the nonrelaxed version promises the tighter bounds of nodes in a B&P. In addition, in some VRP variants such as the team orienteering problem [10], the relaxation should be avoided because it brings a malfunction [2, 11] to a B&P. In this paper, we also do not relax the elementary path constraint.

Among the various types of the resource constraints, this research considers the most representative ones, namely, the vehicle capacity constraint and the vertex time window constraint. The RCESPP can be defined as follows. Let a weighted digraph $G = (V, A)$ be given, where V and A denote sets of vertices and arcs, respectively. Each vertex $v_i \in V$ has a demand d_i and a vehicle has capacity Q. A vehicle should depart from the source $v_0 \in V$ and end at the sink $v_{n+1} \in V$. A vehicle can visit a subset of vertices only if the sum of demands of the visited vertices does not exceed Q. A vehicle takes travelling time $t_{i,j}$ to traverse an arc $(i, j) \in A$ and service time u_i to serve $v_i \in V$. A vehicle can visit $v_i \in V$ only between its time windows $[e_i, l_i]$ and must wait until e_i if the vehicle arrives before e_i. A vehicle pays the travel cost $d_{i,j}$ when it traverses $(i, j) \in A$ and collects the prize g_j when it visits $v_j \in V$. From now on, we denote the cost of an arc

$(i, j) \in A$ as $c_{i,j} = d_{i,j} - g_j$ for simplicity. Let P be a set of all possible paths from v_0 to v_{n+1} that satisfies the vehicle capacity and vertex time window constraints. Let $c(p)$ represent the cost of a path $p \in P$. The optimal path of the RCESPP or p^1 can be found by solving the following problem:

$$p^1 = \arg\min_{p \in P} c(p). \tag{1}$$

The resource-constrained kth elementary shortest path problem (RCKESPP) is a generalization of the RCESPP that aims to determine the kth shortest path when a set of $k-1$ shortest paths or $K = \{p^1, p^2, \ldots, p^{k-1}\}$ is given, where p^i represents the ith shortest path. The optimal path of the RCKESPP or p^k can be found by solving the following problem:

$$p^k = \arg\min_{p \in P \setminus K} c(p). \tag{2}$$

The RCKESPP has many practical applications. Göthe-Lundgren et al. [12] used a constraint generation method to solve the vehicle routing game, in which the vehicle routing cost is allocated to the customers as fairly as possible. The separation problem of the method can be viewed as the RCKESPP. Liu and Ramakrishnan [13] viewed the RCKESPP as a quality of service (QOS) routing problem in the telecommunication industry. van der Zijpp and Catalano [14] viewed the RCKESPP as a path enumeration problem in the transportation industry. Shi [15] viewed the RCKESPP as a robust and stable routing problem in an automated storage and retrieval system. Boussier et al. [11] implicitly showed that the RCKESPP could emerge as a pricing problem in the column generation method for the team orienteering problem. However, Boussier et al. [11] chose the RCESPP as a pricing problem instead of the RCKESPP because they believed that the latter was more difficult to solve than the former.

In an instance with a reasonable number of vertices, enumerating every member of P is nearly impossible. Thus, Lawler's [16] algorithm has been used to solve the RCKESPP [14, 15], which does not require to fill in P. To the best of our knowledge, the RCKESPP has been solved most efficiently by using Lawler's algorithm. This paper proposes a new approach to the RCKESPP named multiple-starting-path (MSP) approach in Section 2. Section 3 reports the computational results which indicate that MSP approach outperforms Lawler's algorithm. Section 4 concludes the paper.

2. MSP Approach

This section begins with introducing a new generalization of the RCESPP, the RCESPP with multiple-starting-paths (RCESPP-MSP). The RCESPP-MSP aims to determine the shortest path from the given starting paths to v_{n+1} that satisfies the resource constraints. Let $S = \{s_1, \ldots, s_m\}$ be a set of m starting paths, where each $s_i \in S$ represents a starting path from v_0 to $v_i \in V$ (this paper uses an alphabet "p" to denote a path from v_0 to v_{n+1} and "s" to denote a path from v_0 to $v_i \in V$). Let p_i^* be the shortest path from $s_i \in S$ to v_{n+1} that satisfies the resource constraints. The optimal path of the RCESPP-MSP or p^* can be found by solving the following problem:

$$p^* = \arg\min_{s_i \in S} c(p_i^*). \tag{3}$$

In the following, Section 2.1 describes the DP algorithm for the RCESPP-MSP and identifies the properties of the RCESPP-MSP. Section 2.2 shows how the RCKESPP is reduced to the RCESPP-MSP.

2.1. RCESPP-MSP. From now on, we refer to the RCESPP-MSP with a set of starting paths S as RCP(S). RCP(S) can be solved by the DP algorithms [2–5] which were originally developed for the RCESPP. For example, the most basic DP algorithm of Feillet et al. [2] can solve RCP(S) as Procedure 1.

In the DP algorithm, a path s from v_0 to $v_i \in V$ is represented as a state with a label (i, c, q, t, E). Each of i, c, q, and t represents the index of the vertex in which the state ends, cost, consumption of vehicle capacity, and time, respectively. $E = (E_1, \ldots, E_n)$ represents the visited vertices of the state, where E_i equals 1 if the state visited $v_i \in V$ and equals 0 otherwise. A state $s = (i, c, q, t, E)$ can be extended to $v_j \in V$ if

$$\begin{gathered} q + d_j \le Q, \\ t + t_{i,j} + u_i \le l_j, \\ E_j \ne 1. \end{gathered} \tag{4}$$

If a state $s = (i, c, q, t, E)$ is extended to $v_j \in V$, then the extended state $s' = (j, c', q', t', E')$ is defined as follows:

$$\begin{gathered} c' = c + c_{i,j}, \\ q' = q + d_j, \\ t' = \max\left(t + t_{i,j} + u_i, e_j\right), \\ E'_k = \begin{cases} 1, & \text{if } k = j \\ E_k, & \text{otherwise,} \end{cases} \\ \forall v_k \in V. \end{gathered} \tag{5}$$

Given two states $s = (i, c, q, t, E)$ and $s'' = (i, c'', q'', t'', E'')$, s dominates s'' if at least one of the following inequalities is strict:

$$\begin{gathered} c \le c'', \\ q \le q'', \\ t \le t'', \\ E_k \le E''_k, \quad \forall v_k \in V. \end{gathered} \tag{6}$$

Given a set of starting states S, DP(S) finds the shortest path from the states in S to v_{n+1}. DP(S) starts by placing each

Procedure DP(S)
$L_i = \varnothing$; $\forall v_i \in V$
$\forall s = (i, c, q, t, E) \in S$
  $L_i = L_i \cup \{s\}$;
  $EFF(L_i)$;
Repeat
  $\forall v_i \in V \setminus v_{n+1}$
    $\forall$ un-extended state $s \in L_i$
      $\forall v_j \in V^+(v_i)$
        if the extension of s to v_j is feasible
          $s' = \text{extend}(s, v_j)$;
          $L_j = L_j \cup \{s'\}$;
          $EFF(L_j)$;
    mark s as the extended state;
Until there is no un-extended state in L_i, $\forall v_i \in V \setminus v_{n+1}$
Return $p^* = \arg\min_{s=(n+1,c,q,t,E)\in L_{n+1}} c$;

Procedure 1

starting state $s = (i, c, q, t, E) \in S$ into the corresponding repository L_i, which stores the states that end at $v_i \in V$. Each state in L_i is extended to other vertices if the extension is feasible. This extension is repeated until no state can be extended feasibly. During the extension, the dominated states are deleted and only the dominant states are kept. $V^+(v_i) = \{v_j \in V : (i, j) \in A\}$ represents a set of successors from $v_i \in V$. The function extend(s, v_j) extends states $s = (i, c, q, t, E)$ to v_j and then returns the extended state. The function EFF(L_i) deletes the dominated states in L_i.

Similarly, the other DP algorithms [3, 5, 9] can solve RCP(S). Among them, we use the state-of-the-art algorithm of Baldacci et al. [4] to solve RCP(S). Here, we describe the algorithm briefly and readers are recommended to see Baldacci et al. [4] for details. Baldacci et al. [4] introduced *ng*-route relaxation and used the *ng*-route relaxation to calculate the lower bound of a state. A state is deleted if its lower bound is worse than the incumbent solution. The *ng*-route relaxation requires an incumbent solution to delete unpromising states. The incumbent solution can be obtained by a heuristic algorithm with cheap computational cost [8, 11, 17]. This paper calculates an incumbent solution by a simple DP algorithm which is the same as Procedure 1 except that its dominance rule does not consider the condition $E_k \leq E''_k$, $\forall v_k \in V_C$. The simple DP algorithm allows more domination between states but it may delete the states which yield the shortest path.

The RCESPP-MSP presents the following properties.

Property 1. Suppose that sets of starting paths $S_1 = \{s_{1,1}, \ldots, s_{1,n_1}\}$, $S_2 = \{s_{2,1}, \ldots, s_{2,n_2}\}$, and $S_3 = S_1 \cup S_2$ are given, where each of $s_{1,i} \in S_1$ and $s_{2,j} \in S_2$ represents a starting path from v_0 to $v_k \in V$ that satisfies the resource constraints. Let the optimal solutions of RCP(S_1), RCP(S_2), and RCP(S_3) be $p^*_{S_1}$, $p^*_{S_2}$, and $p^*_{S_3}$, respectively. Then, $c(p^*_{S_3}) = \min(c(p^*_{S_1}), c(p^*_{S_2}))$.

Proof. (i) For each starting path $s_{1,i} \in S_1$, let $p^*_{1,i}$ represent the shortest path from $s_{1,i}$ to v_{n+1} that satisfies the resource constraints. Similarly, let $p^*_{2,i}$ represent the shortest path from $s_{2,i} \in S_2$ to v_{n+1} that satisfies the resource constraints.

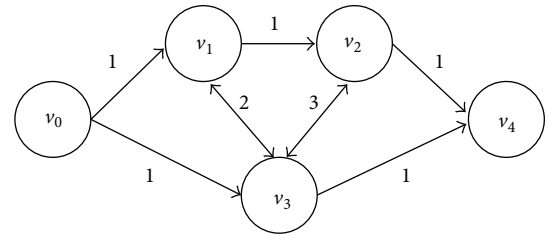

Figure 1: An example graph.

(ii) Based on the definition of RCESPP-MSP, the following equalities hold:

$$c\left(p^*_{S_1}\right) = \min\left(c\left(p^*_{1,1}\right), \ldots, c\left(p^*_{1,n_1}\right)\right) \tag{7}$$

$$c\left(p^*_{S_2}\right) = \min\left(c\left(p^*_{2,1}\right), \ldots, c\left(p^*_{2,n_2}\right)\right) \tag{8}$$

$$c\left(p^*_{S_3}\right) = \min\left(c\left(p^*_{1,1}\right), \ldots, c\left(p^*_{1,n_1}\right), c\left(p^*_{2,1}\right), \ldots, c\left(p^*_{2,n_2}\right)\right). \tag{9}$$

(iii) Therefore, by using (7) and (8), $\min(c(p^*_{S_1}), c(p^*_{S_2}))$ can be expressed as follows:

$$\min\left(\min\left(c\left(p^*_{1,1}\right), \ldots, c\left(p^*_{1,n_1}\right)\right), \min\left(c\left(p^*_{2,1}\right), \ldots, c\left(p^*_{2,n_2}\right)\right)\right). \tag{10}$$

(iv) The right-hand side of (9) is the same as (10). Therefore, $c(p^*_{S_3}) = \min(c(p^*_{S_1}), c(p^*_{S_2}))$.

Property 1 shows that RCP(S_3) is the same with the problem that tries to determine the shorter path between the optimal paths of RCP(S_1) and RCP(S_2). □

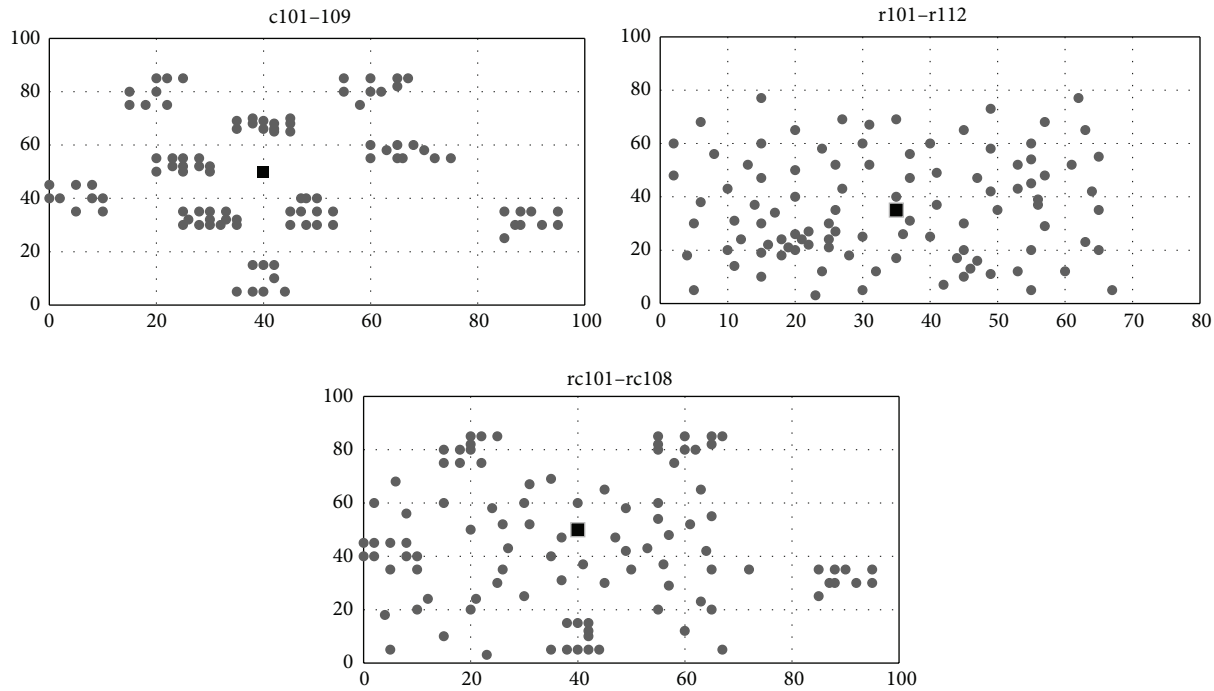

Figure 2: Geometric view of the benchmark instances.

Table 1: Subproblems of Lawler's algorithm when $k = 4$.

$s \in A(K)$	$B(s, K)$	$S(s, K)$	Shortest path	Cost
(v_0)	$\{v_1, v_3\}$	No feasible path	No path	—
(v_0, v_3)	$\{v_4\}$	$\{(v_0, v_3, v_1), (v_0, v_3, v_2)\}$	(v_0, v_3, v_2, v_4)	5
(v_0, v_1)	$\{v_2, v_3\}$	No feasible path	No path	—
(v_0, v_1, v_2)	$\{v_4\}$	$\{(v_0, v_1, v_2, v_3)\}$	$(v_0, v_1, v_2, v_3, v_4)$	6
(v_0, v_1, v_3)	$\{v_4\}$	$\{(v_0, v_1, v_3, v_2)\}$	$(v_0, v_1, v_3, v_2, v_4)$	7

Property 2. Suppose that the RCESPP-MSP is solved by the DP algorithm and let each of t_1, t_2, and t_3 represent the computational time of the DP algorithm to solve RCP(S_1), RCP(S_2), and RCP($S_3 = S_1 \cup S_2$), respectively. Then, $t_3 \leq t_1 + t_2$.

Proof. (i) A set of dominant states remains when the DP algorithm terminates. Let each of D_1, D_2, and D_3 represent a set of remaining dominant states when the DP algorithm terminates after solving RCP(S_1), RCP(S_2), and RCP(S_3), respectively.

(ii) Domination between D_1 and D_2 may or may not occur. When such domination occurs, $|D_3| < |D_1| + |D_2|$ and $|D_3| = |D_1| + |D_2|$ otherwise. Therefore, $|D_3| \leq |D_1| + |D_2|$.

(iii) The computational time of DP algorithm is mainly decided by the number of generated states as shown in [2, 3, 5]. Therefore, $t_3 \leq t_1 + t_2$.

Property 2 indicates that RCP(S_3) can be solved faster than the problem that tries to determine the shorter path between the optimal paths of RCP(S_1) and RCP(S_2). □

2.2. Reduction of RCKESPP to RCESPP-MSP. This subsection begins with describing Lawler's algorithm with the example graph in Figure 1. The graph constitutes a set of five vertices $\{v_0, v_1, v_2, v_3, v_4\}$, where v_0 and v_4 represent the origin and the sink, respectively. The shortest path from v_0 to v_4 is $p_1 = (v_0, v_3, v_4)$, while the second and third shortest paths are $p_2 = (v_0, v_1, v_2, v_4)$ and $p_3 = (v_0, v_1, v_3, v_4)$, respectively. Suppose that the fourth shortest path is searched by Lawler's algorithm when $K = \{p_1, p_2, p_3\}$ is given. The resource constraints are not considered in this example for simplicity, but one may easily apply the constraints to the graph.

The two paths (v_0) and (v_0, v_3) from p_1 and the three paths (v_0), (v_0, v_1), and (v_0, v_1, v_2) from p_2 can be generated. Let $A(K)$ represent a set of the paths which are generated from the paths in K. In this graph, $A(K)$ is expressed as $\{(v_0), (v_0, v_3), (v_0, v_1), (v_0, v_1, v_2), (v_0, v_1, v_3)\}$. (v_0, v_1) and p_2 are connected by v_2. Similarly, (v_0, v_3) and p_1 are connected by v_4. Let $B(s, K)$ represent a set of connecting vertices between $s \in A(K)$ and K. For example, $B((v_0, v_1), K) = \{v_2, v_3\}$ and $B((v_0, v_3), K) = \{v_4\}$. Given a path s, let $v_{\text{end}}(s)$ represent the last vertex, in which s ends. For example, $v_{\text{end}}((v_0, v_1)) = v_1$ and $v_{\text{end}}((v_0, v_3)) = v_3$. Each path $s \in A(K)$ can be extended to other vertices as long as the extension satisfies the resource constraints. Let $S(s, K)$ represent a set of feasibly extended paths from s to each vertex $v \in V^+(v_{\text{end}}(s)) \setminus B(s, K)$. For example, $S((v_0, v_1), K) = \varnothing$ and $S((v_0, v_3), K) = \{(v_0, v_3, v_1), (v_0, v_3, v_2)\}$.

Table 2: Computational result for $k = 2, 5$.

	$k = 2$						$k = 5$					
	Lawler		MSP		Gap (%)		Lawler		MSP		Gap (%)	
	Time	States	Time	States	Time	States	Time	States	Time	States	Time	States
c101	0.22	262015	0.05	51620	79.3	80.3	0.30	328115	0.05	51888	84.2	84.2
c102	1.53	1674557	0.27	292153	82.1	82.6	1.97	2238064	0.26	288195	86.6	87.1
c103	5.42	5208681	1.29	1034255	76.3	80.1	13.71	13655084	1.28	1027669	90.7	92.5
c104	14.31	11352655	3.58	2264392	75.0	80.1	24.12	20139051	3.73	2304822	84.5	88.6
c105	0.31	367544	0.06	71265	79.7	80.6	0.39	471589	0.07	71514	83.5	84.8
c106	0.41	502159	0.09	100467	78.0	80.0	0.50	611395	0.09	100752	81.7	83.5
c107	0.40	480488	0.09	97047	78.3	79.8	0.52	637315	0.09	97175	83.4	84.8
c108	0.72	877250	0.17	180971	76.8	79.4	1.47	1864824	0.16	185990	88.9	90.0
c109	1.26	1519472	0.27	309394	78.3	79.6	2.53	3177807	0.28	320461	89.0	89.9
r101	0.06	66944	0.02	20063	68.3	70.0	0.16	174187	0.02	20387	86.6	88.3
r102	11.40	3626432	2.09	938016	81.7	74.1	13.84	5065526	2.10	938592	84.8	81.5
r103	8.80	5412423	2.20	1195766	75.0	77.9	15.66	11033799	2.53	1301636	83.8	88.2
r104	35.32	11639745	17.01	4586566	51.8	60.6	40.27	22129546	8.01	3089736	80.1	86.0
r105	0.25	268513	0.06	55404	76.0	79.4	0.54	587717	0.06	55799	88.3	90.5
r106	14.28	4313228	3.49	1419126	75.5	67.1	20.98	7196699	4.43	1566986	78.9	78.2
r107	18.37	7645776	4.04	1863475	78.0	75.6	28.68	14166896	4.00	1854919	86.0	86.9
r108	22.13	10614957	4.61	2254591	79.2	78.8	28.20	16276837	10.26	3465735	63.6	78.7
r109	0.81	839422	0.18	160849	77.8	80.8	1.63	1816296	0.18	164546	88.8	90.9
r110	5.04	3185193	0.91	674064	81.9	78.8	9.06	7991178	0.92	680420	89.9	91.5
r111	36.20	6664411	3.88	1762656	89.3	73.6	38.01	11011212	3.91	1764017	89.7	84.0
r112	11.51	6618433	7.31	2714618	36.5	59.0	22.79	14307286	5.22	2278570	77.1	84.1
rc101	0.16	171470	0.04	40719	73.9	76.3	0.34	364195	0.04	40803	87.5	88.8
rc102	0.73	821774	0.17	164979	77.2	79.9	1.77	2026238	0.17	170627	90.5	91.6
rc103	2.53	2502612	0.49	454496	80.8	81.8	4.24	4720830	0.49	454972	88.6	90.4
rc104	32.02	9320068	1.86	1303477	94.2	86.0	33.66	11164378	1.57	1197120	95.3	89.3
rc105	0.43	473190	0.11	110540	74.1	76.6	0.99	1141481	0.11	110565	89.1	90.3
rc106	0.54	586291	0.14	131140	74.0	77.6	0.96	1111932	0.14	131497	85.5	88.2
rc107	1.62	1634032	0.45	402166	72.2	75.4	3.16	3363444	0.46	406271	85.6	87.9
rc108	3.15	3112749	0.82	717401	73.8	77.0	6.92	7309195	0.84	718232	87.9	90.2

For each path $s \in A(K)$, Lawler's algorithm solves RCP($S(s, K)$) as shown in Table 1. Among the shortest paths, Lawler's algorithm finds the one with the minimum cost. (v_0, v_3, v_2, v_4) is the path with the minimum cost and is hence identified as the fourth shortest path. For each path $s_i \in A(K)$, let S_i represent $S(s_i, K)$ and $S^{\text{All}} = \cup_{s_i \in A(K)} S_i$. For each $S_i \in S^{\text{All}}$, let $p^*_{S_i}$ represent the optimal path of RCP(S_i). Lawler's algorithm solves the RCKESPP by solving the following problem:

$$p^k = \arg\min_{S_i \in S^{\text{All}}} c\left(p^*_{S_i}\right). \tag{11}$$

Property 1 indicates that problem (11) reduces to RCP(S^{All}). Therefore, the optimal path of the RCKESPP or p^k can be found by solving RCP(S^{All}). Property 2 shows that RCP(S^{All}) can be solved faster than problem (11). In other words, MSP approach can solve the RCKESPP in less time than Lawler's algorithm. In summary, MSP approach is shown in Procedure 2. MSP(K) makes S^{All} based on K and then solves RCP(S^{All}) using the DP algorithm. Then, the kth shortest path or p^k is returned.

MSP(K) calculates S^{All} for the example problem in Figure 1 as follows:

$$S^{\text{ALL}} = \{(v_0, v_3, v_1), (v_0, v_3, v_2), (v_0, v_1, v_2, v_3), (v_0, v_1, v_3, v_2)\}. \tag{12}$$

Then, the shortest path from the paths in S^{All} to v_4 is determined using the DP algorithm. In the first steps of the DP algorithm, the path (v_0, v_1, v_3, v_2) will be deleted because it is dominated by (v_0, v_3, v_2). Afterwards, the fourth shortest path or (v_0, v_3, v_2, v_4) is found by the DP algorithm that considers $\{(v_0, v_3, v_1), (v_0, v_3, v_2), (v_0, v_1, v_2, v_3)\}$ as a set of starting paths.

3. Computational Results

The well-known vehicle routing problem with time windows instance set of Solomon [18] is used in the computational

Procedure MSP(K)
$S^{\text{All}} = \varnothing$;
calculate $A(K)$ first then calculate $B(s, K)$ and $S(s, K)$ for each $s \in A(K)$;
$\forall s \in A(K)$
  $S^{\text{All}} = S^{\text{All}} \cup S(s, K)$;
$p^k = \text{DP}\left(S^{\text{All}}\right)$
Return p^k;

Procedure 2

Table 3: Computational result for $k = 10, 20$.

	$k = 10$						$k = 20$					
	Lawler		MSP		Gap (%)		Lawler		MSP		Gap (%)	
	Time	States	Time	States	Time	States	Time	States	Time	States	Time	States
c101	0.44	485261	0.05	52668	89.0	89.1	0.58	713966	0.05	53603	91.2	92.5
c102	2.14	2515472	0.26	289265	87.6	88.5	3.29	4018492	0.27	291578	91.9	92.7
c103	18.52	18809416	1.28	1035057	93.1	94.5	21.83	23252746	1.30	1043985	94.0	95.5
c104	52.08	47250175	3.95	2386036	92.4	95.0	83.03	80406120	3.85	2355629	95.4	97.1
c105	0.81	990708	0.07	72480	91.9	92.7	1.08	1348727	0.07	74274	93.5	94.5
c106	0.71	888745	0.09	101630	86.8	88.6	1.01	1285671	0.09	102617	90.8	92.0
c107	0.92	1138227	0.09	98226	90.4	91.4	1.42	1791060	0.09	100349	93.5	94.4
c108	2.43	3139895	0.17	188067	93.2	94.0	3.06	4017208	0.17	190044	94.4	95.3
c109	4.53	5763260	0.29	327949	93.6	94.3	6.48	8428446	0.30	334841	95.4	96.0
r101	0.20	220868	0.02	20631	88.8	90.7	0.27	308245	0.02	21113	91.5	93.2
r102	22.09	10044289	2.10	940377	90.5	90.6	28.29	14080025	2.11	943007	92.5	93.3
r103	22.69	16937684	3.46	1560465	84.8	90.8	47.52	34124077	3.42	1559880	92.8	95.4
r104	60.28	38761765	10.12	3519272	83.2	90.9	137.14	67258510	11.46	3478336	91.6	94.8
r105	0.71	804350	0.06	56411	91.3	93.0	1.29	1488830	0.07	57934	94.9	96.1
r106	32.91	11726986	4.37	1569976	86.7	86.6	53.75	20178129	5.00	1652562	90.7	91.8
r107	43.81	28328004	4.02	1857682	90.8	93.4	64.43	42059268	4.67	2029027	92.8	95.2
r108	29.41	17564810	8.30	3098793	71.8	82.4	69.22	42265459	8.29	3104067	88.0	92.7
r109	3.08	3448253	0.19	169998	93.8	95.1	5.44	6153499	0.20	175407	96.3	97.1
r110	13.43	12803914	0.91	682379	93.2	94.7	18.85	18872901	0.90	679098	95.2	96.4
r111	40.88	12806004	4.63	1932727	88.7	84.9	42.23	15719756	5.54	2116709	86.9	86.5
r112	28.63	18875159	6.01	2465366	79.0	86.9	63.61	37552120	7.27	2710032	88.6	92.8
rc101	0.60	678727	0.05	41805	92.5	93.8	0.90	1052775	0.05	42887	94.7	95.9
rc102	2.75	3145387	0.17	171178	93.7	94.6	4.41	5217034	0.18	173211	96.0	96.7
rc103	5.38	6218635	0.49	456220	90.9	92.7	7.84	9202713	0.50	467826	93.6	94.9
rc104	34.92	12895802	1.71	1242992	95.1	90.4	39.97	18859015	1.96	1340840	95.1	92.9
rc105	1.60	1863337	0.11	112238	93.0	94.0	1.81	2134700	0.12	113400	93.6	94.7
rc106	1.53	1815310	0.14	131787	90.8	92.7	1.95	2374017	0.15	133691	92.5	94.4
rc107	4.97	5621602	0.46	406742	90.8	92.8	6.57	7717624	0.46	407187	93.0	94.7
rc108	12.87	13877100	0.87	746921	93.2	94.6	22.14	24324543	0.90	756113	95.9	96.9

experiment. Each instance has a complete graph with 102 vertices, including the origin and sink nodes. Each instance has a vehicle with a capacity of 200. The instance set is classified as R, C, and RC depending on how the vertices are geographically located. The locations of the vertices are randomly distributed in Type R instances and are clustered in Type C instances. In Type RC instances, some vertices are distributed randomly whereas others are clustered. The instances are grouped as r101–r112, c101–c109, and rc101–rc108 depending on their type. The tightness of the time windows is the only difference among the same types of instances. Figure 2 shows the geometric view of the benchmark instances, where a square represents the depot (the origin and the sink) and circles represent the customers.

Vertex v_i has its own coordination (x_i, y_i) and prize g_i. The travel time from $v_i \in V$ to $v_j \in V$ is calculated as $t_{ij} = \lfloor \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \rfloor$ and the cost as $c_{i,j} = t_{i,j} - g_j$. This

paper sets g_i as integers from 0 to 20 to obtain a reasonable percentage of negative arcs. The integers are set following Righini and Salani [5].

Computational tests were performed using an Intel Dual core E7300 2.66 GHZ 2.67 GHZ PC with 4 GB RAM. The most recent algorithm for the RCKESPP was proposed by Shi [15], who followed the procedure of Lawler [16]. Therefore, this paper compares MSP approach with Lawler's algorithm.

Tables 2 and 3 show the computational times and the numbers of generated states by Lawler's algorithm and MSP approach for the instances when k is 2, 5, 10, and 20. The computational time is measured in seconds. The columns headed with Gap(%) give the proposed approach's percentage of the improvement over Lawler's. These tables show that MSP approach consistently generates fewer states and thus consumes less computational time for every instance and k value. Lawler's algorithm generates more states and consumes more computational time as k increases, while MSP approach shows consistent performance regardless of k.

4. Conclusion

Lawler's algorithm has been known as the most efficient way for solving the RCKESPP. This algorithm can solve the RCKESPP by solving $(k-1)(|V|-1)$ times of the RCESPP-MSP in the worst case [15]. Therefore, the RCKESPP has been believed as a more difficult problem than the RCESPP. This paper proposes a new approach for solving the RCKESPP named MSP approach which reduces the RCKESPP to the RCESPP-MSP. This paper presents computational result that shows how MSP approach outperforms Lawler's algorithm in every instance and every k value.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgment

This work was supported by the National Research Foundation of Korea Grant funded by the Korean Government (NRF-2012R1A1A2005243).

References

[1] M. Dror, "Note on the complexity of the shortest path models for column generation in VRPTW," *Operations Research*, vol. 42, no. 5, pp. 977–978, 1994.

[2] 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*, vol. 44, no. 3, pp. 216–229, 2004.

[3] G. Righini and M. Salani, "Symmetry helps: bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints," *Discrete Optimization*, vol. 3, no. 3, pp. 255–273, 2006.

[4] R. Baldacci, N. Christofides, and A. Mingozzi, "An exact algorithm for the vehicle routing problem based on the set partitioning formulation with additional cuts," *Mathematical Programming*, vol. 115, no. 2, pp. 351–385, 2008.

[5] G. Righini and M. Salani, "New dynamic programming algorithms for the resource constrained elementary shortest path problem," *Networks*, vol. 51, no. 3, pp. 155–170, 2008.

[6] M. Desrochers, J. Desrosiers, and M. Solomon, "A new optimization algorithm for the vehicle routing problem with time windows," *Operations Research*, vol. 40, no. 2, pp. 342–354, 1992.

[7] G. Desaulniers, F. Lessard, and A. Hadjar, "Tabu search, partial elementarity, and generalized k-path inequalities for the vehicle routing problem with time windows," *Transportation Science*, vol. 42, no. 3, pp. 387–404, 2008.

[8] A. Chabrier, "Vehicle routing problem with elementary shortest path based column generation," *Computers & Operations Research*, vol. 33, no. 10, pp. 2972–2990, 2006.

[9] R. Baldacci, A. Mingozzi, and R. Roberti, "New route relaxation and pricing strategies for the vehicle routing problem," *Operations Research*, vol. 59, no. 5, pp. 1269–1283, 2011.

[10] H. Tae and B.-I. Kim, "A branch-and-price approach for the team orienteering problem with time windows," *International Journal of Industrial Engineering: Theory, Applications and Practice*. In press.

[11] S. Boussier, D. Feillet, and M. Gendreau, "An exact algorithm for team orienteering problems," *4OR*, vol. 5, no. 3, pp. 211–230, 2007.

[12] M. Göthe-Lundgren, K. Jörnsten, and P. Värbrand, "On the nucleolus of the basic vehicle routing game," *Mathematical Programming*, vol. 72, no. 1, pp. 83–100, 1996.

[13] G. Liu and K. G. Ramakrishnan, "A* prune: an algorithm for finding K shortest paths subject to multiple constraints," in *Proceedings of the 20th Annual Joint Conference of the IEEE Computer and Communications Societies*, pp. 743–749, Anchorage, Alaska, USA, April 2001.

[14] N. J. van der Zijpp and S. F. Catalano, "Path enumeration by finding the constrained K-shortest paths," *Transportation Research Part B: Methodological*, vol. 39, no. 6, pp. 545–563, 2005.

[15] N. Shi, "K constrained shortest path problem," *IEEE Transactions on Automation Science and Engineering*, vol. 7, no. 1, pp. 15–23, 2010.

[16] E. L. Lawler, "A procedure for computing the K best solutions to discrete optimization problems and its application to the shortest path problem," *Management Science*, vol. 18, no. 7, pp. 401–405, 1972.

[17] H. Tae and B.-I. Kim, "Dynamic programming approach for prize colleting travelling salesman problem with time windows," *IE Interfaces*, vol. 24, no. 2, pp. 112–118, 2011.

[18] M. M. Solomon, "Algorithms for the vehicle routing and scheduling problems with time window constraints," *Operations Research*, vol. 35, no. 2, pp. 254–265, 1987.