Elementary Resource Constrained Shortest Path Problem (ERCSPP)

Matheus Diógenes Andrade
matheusdiogenesandrade@gmail.com
Fábio Luiz Usberti
fusberti@ic.unicamp.br
Rafael Kendy Arakaki
rafaelkendyarakaki@gmail.com

Institute of Computing - University of Campinas

2021

Topics

---

[1] Dominique Feillet et al. "An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems". In: *Networks* 44.3 (2004), pp. 216–229. DOI: 10.1002/net.20033.

[2] Giovanni Righini and Matteo Salani. "Symmetry helps: Bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints". In: *Discrete Optimization* 3.3 (2006). Graphs and Combinatorial Optimization, pp. 255–273. ISSN: 1572-5286. DOI: https://doi.org/10.1016/j.disopt.2006.05.007. URL: https://www.sciencedirect.com/science/article/pii/S1572528606000417.

Problem definition

Problem definition

Let

- $D(V = \{s\} \cup V^+ \cup \{t\}, A)$ be the digraph, where
  - s is the source node; and
  - t is the target node.
- $c_a \in \mathbb{R}$ be the arc $a \in A$ cost;
- R be the set of resources;
- $d_a^r \in \mathbb{R}$ be the metric resource $r \in R$ consumption of the arc $a \in A$;
- $w_i^r = [b_i^r, e_i^r]$ be the resource $r \in R$ window of the node $i \in V$.

Problem definition

Let

- P be an elementary resource constrained s-t-path in D; and
- S_i^r be the resource $r \in R$ consumption of node $i \in V(P)$, such that $\forall (i,j) \in A(P)(\forall r \in R(S_j^r = \max\{S_i^r + d_{ij}^r, b_j^r\} \leqslant e_j^r))$.

The ERCSPP consists in finding the shortest elementary resource constrained s-t-path P in D.

Problem definition

$c_a : (d_a^1, d_a^2)$

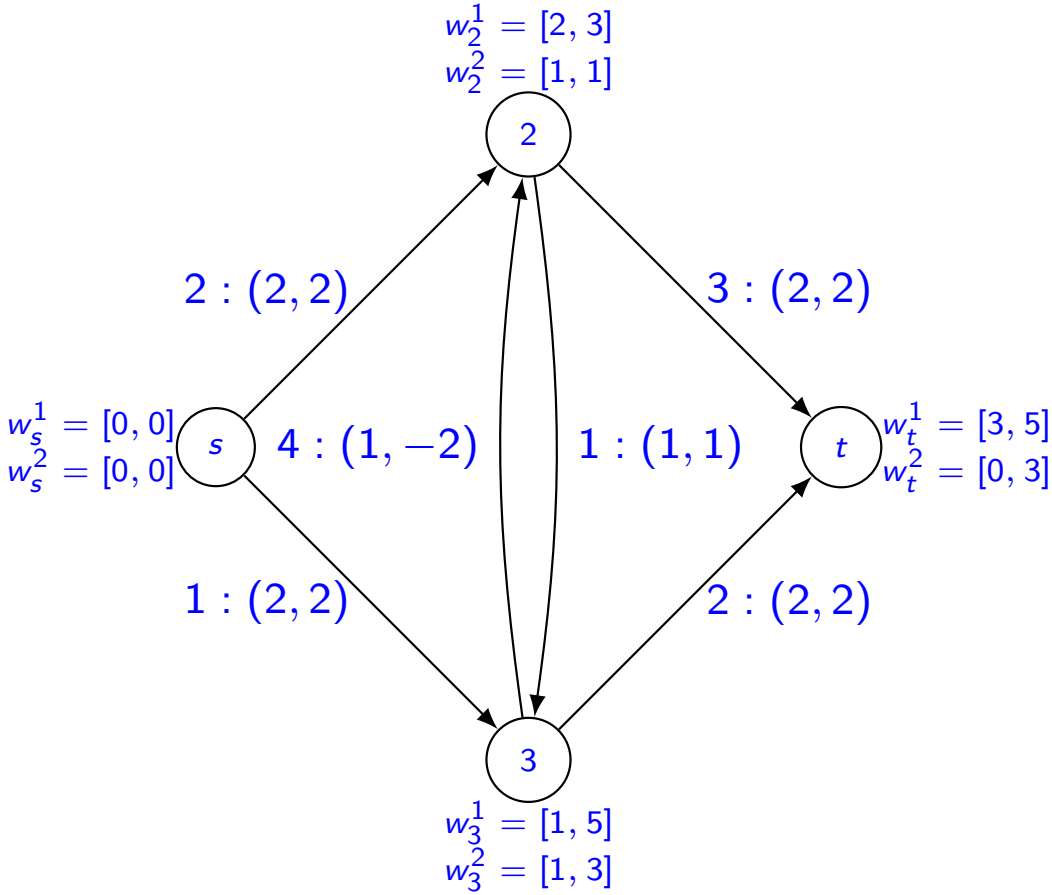

Figure: A ERCSPP instance digraph example.

Problem definition

$c_a : (d_a^1, d_a^2)$

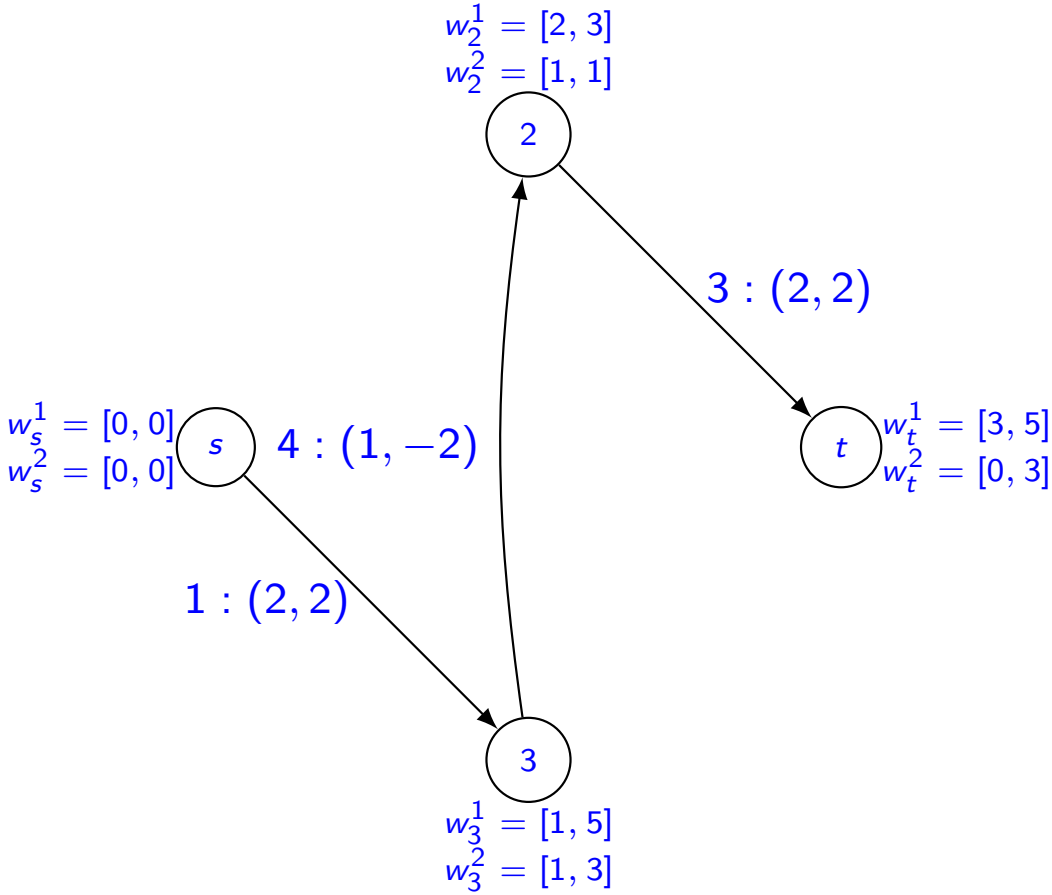

Figure: The solution of the previous instance.

Mono-directional algorithm[3]

[3] Dominique Feillet et al. "An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems". In: *Networks* 44.3 (2004), pp. 216–229. DOI: 10.1002/net.20033.

Properties

Properties

Node state/label I

Let

- $i \in V$ be a node in D;
- P_i be an elementary resource constrained s-i-path in D; and
- $S_i = (L_i = (l_i^1, ..., l_i^{|R|}), C_i)$ be the state or label of node i in P_i, where:
  - $l_i^r \in \mathbb{R}$ is the amount of resource $r \in R$ consumed by P_i; and
  - $C_i \in \mathbb{R}$ is the cost of path P_i.

Properties

Node state/label I

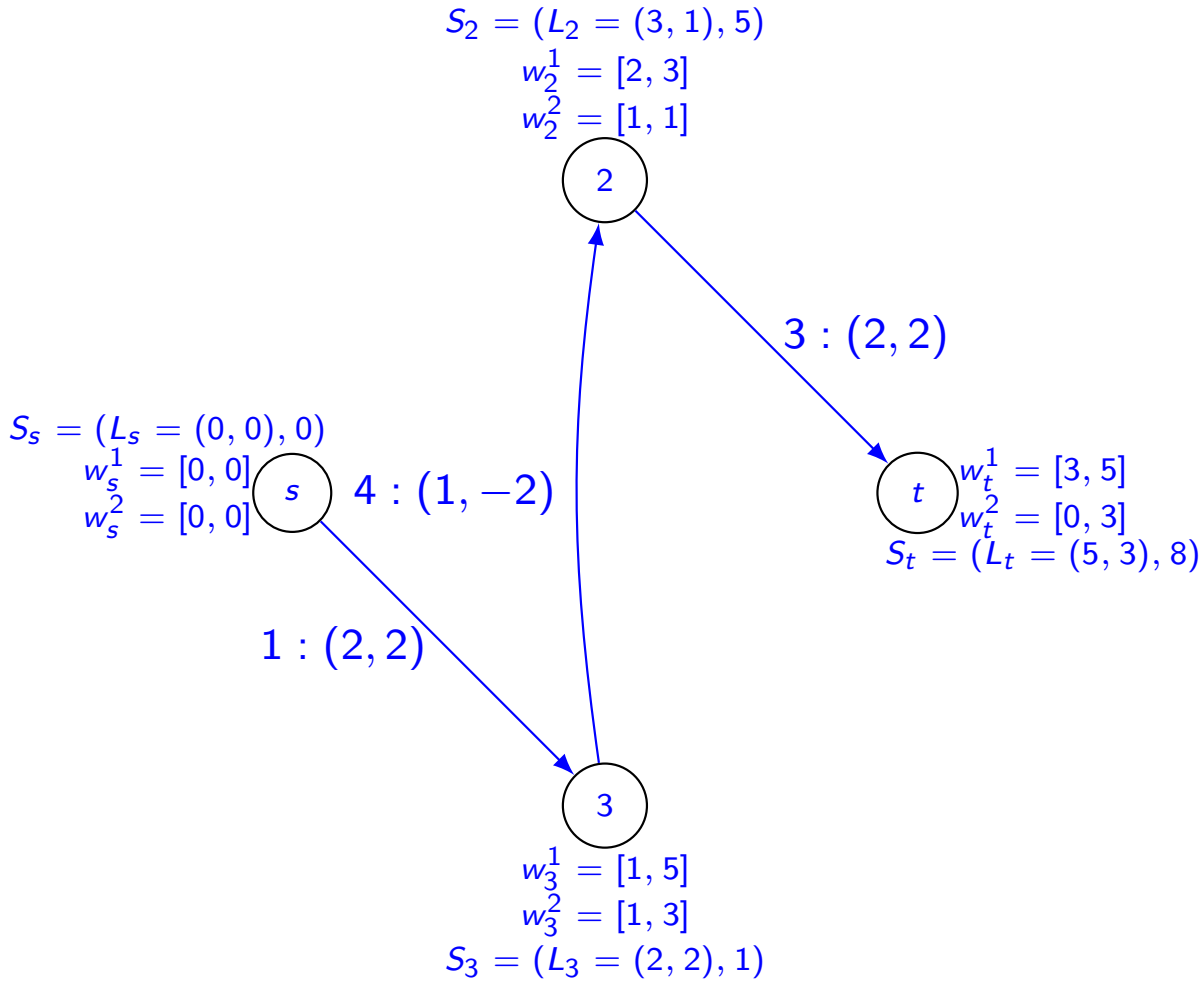

Figure: P_s, P_3, P_2, and P_t.

Properties

Dominance I

Let

- P_i and P_i' be two distinct elementary resource constrained s-i-paths in D; and
- S_i and S_i' be their respective labels.

Definition

P_i dominates $P_i' \leftrightarrow S_i \leqslant S_i'$.

Properties

Node state/label II

Let

- $L_i = (l_i^1, ..., l_i^{|R|}, s_i, v_i^1, ..., v_i^{|V|})$ be the the new resources state/label, where:
  - $s_i = \sum_{j=1}^{|V|} v_i^j = |V(P_i)|$ is the number of nodes visited by P_i; and
  - $v_i^j \in \mathbb{B}$ is equals to 1 if node $j \in V(P_i)$, i.e., j is visited by P_i, such that $j \in V$, and 0 otherwise.

Properties

Node state/label II

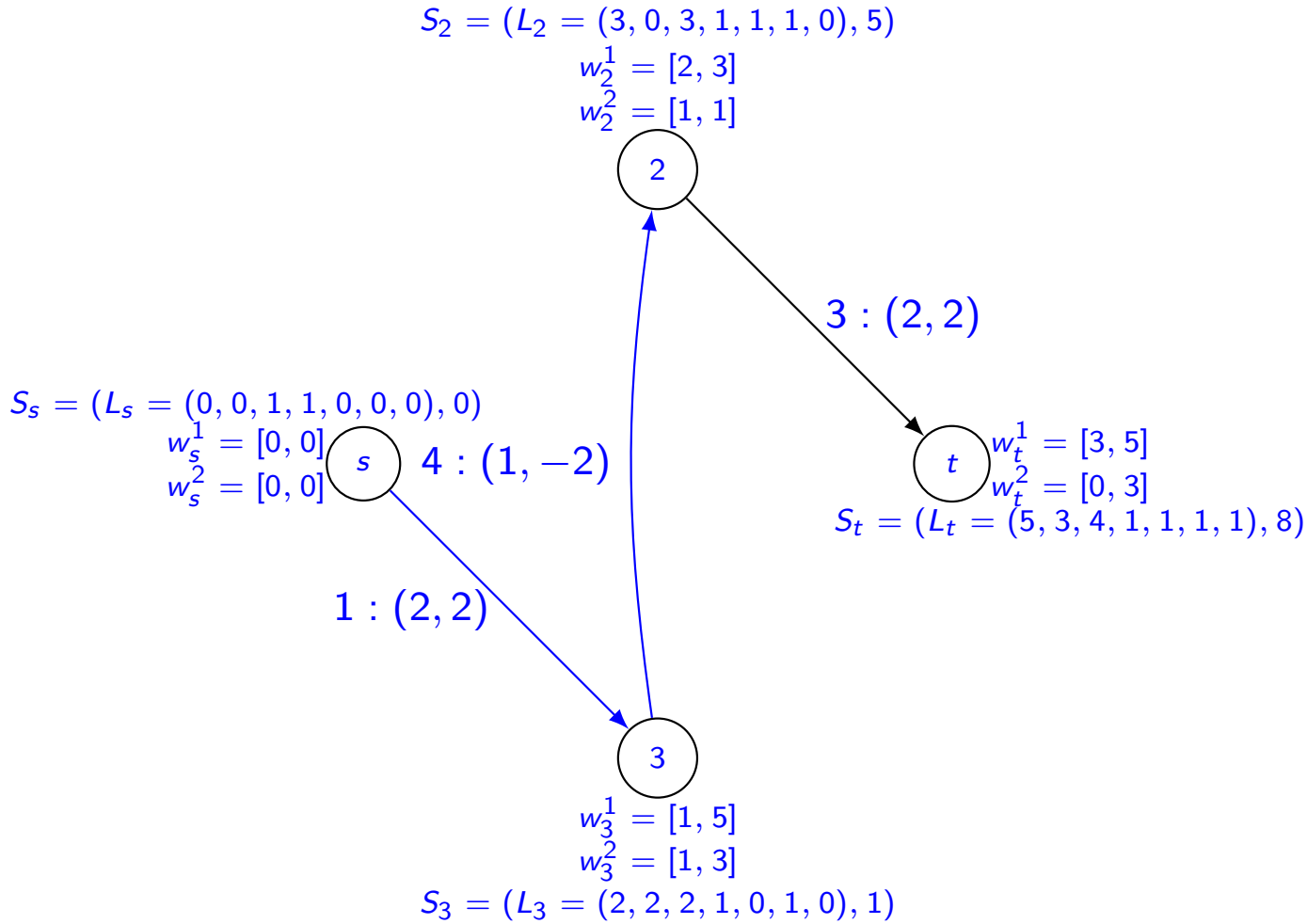

Figure: P_s, P_3, P_2, and P_t with new labels.

Properties

Dominance II

Let

- P_i and P_i' be two distinct elementary paths from s to $i \in V$ in D; and
- S_i and S_i' be the paths respective labels.

Definition

P_i dominates $P_i' \leftrightarrow S_i \leqslant S_i'$.

We can fastly check if $A = (v_i^1, ..., v_i^{|V|}) \leqslant B = (v_i^{1'}, ..., v_i^{|V|'})$ by considering A and B as bitsets and checking whether $(v_i^k \implies v_i^{k'}) \iff (\neg v_i^{k'} \vee v_i^k)$, i.e. $(A \implies B) \iff (\neg B \vee A)$, $\forall k \in \{1, ..., |V|\}$. Hence, check whether a label dominates other can be done in $O(|R|)$.

Properties

Unreachable nodes

Definition

$k \in V$ is said to be unreachable by $P_i \rightarrow k \in V(P_i) \vee$
$\exists r \in R(l_i^r + d_{ik}^r > e_k^r)$.

Note that, if $k \in V$ is unreachable by P_i, then $\nexists$ elementary resource constrainted i-j-path in D, due to the digraph metricity. That is the main reason why we are considering the digraph as complete and metric.

Properties

Node state/label III

Let

- $s_i = \sum_{j=1}^{|V|} v_i^j$ be the number of unreachable nodes by the path P_i at $i \in V(P_i)$; and
- $v_i^j \in \mathbb{B}$ be equals to 1 if node $j \in V$ is unreachable by the path P_i, and 0 otherwise.

Properties

Nondominated paths

Claim: In order to find an ERCSPP optimal solution, suffices to consider only nondominated paths.

Proof: Let's consider two elementary resource constrained s-i-paths P_i and P_i', along with its labels S_i and S_i', such that P_i dominates P_i'. Also, let's consider an arc $(i,j) \in \delta^+(i) : \forall r \in R$ $(l_i'^r + d_{ij}^r \leqslant e_j^r) \wedge v_i'^j = 0$. Note that $\forall r \in R(l_i^r + d_{ij}^r \leqslant e_j^r) \wedge v_i^j = 0 \wedge C_i \leqslant C_i'$. ■

Algorithm

Algorithm

Notations I

Let

- Λ_i be the list of labels on node $i \in V$;
- N be the list of nodes to be processed;
- $E(S_i, j) = (\max\{l_i^r + d_{ij}^r, b_j^r\} : r \in R) \cup (s_i + 1, v_i^1, ..., v_i^j + 1, ..., v_i^{|V|})$ be the function that returns the label resulting from the extension of a path P_i by a node j ($O(|R|)$);
- $$f(S_i, j) = \begin{cases} \text{true}, & \text{if } \forall r \in R(\max\{l_i^r + d_{ij}^r, b_j^r\} \leqslant e_i^r) \wedge \neg v_i^j \\ \text{false}, & \text{otherwise} \end{cases}$$

  be a function that says whether an extension is feasible ($O(|R|)$);
- F_{ij} be the set of labels extended from i to $j \in V$; and

Algorithm

Notations II

Let

- $N_D(\Lambda) = \{S_i \in \Lambda : \nexists S_i' \in \Lambda(S_i' \neq S_i \wedge S_i' \leqslant S_i)\}$ be the function that returns all nondominated labels from Λ.
  - A naïve algorithm would take $O(|\Lambda||R|)$ to check whether a state is nondominated. However, keeping $|R|$ Binary Seach Trees (BSTs) associated with the labels set Λ_i may help us to reduce the time complexity;
  - Let BST_i^r be the BST containing all the resource $r \in R$ consumptions of all labels in Λ_i, $g(BST_i^r, l_i^r)$ be the rightmost position of BST_i^r whose value is equals to or less than l_i^r, and $S(BST_i^{rj})$ be the label associated with the resource consumption at the j-th position of BST_i^r;
  - To check if a label S_i is nondominated it would take $O(min_{r \in R}\{g(BST_i^r, l_i^r)\}|R| + |R|log_2^{|R|})$;
  - Note that the above complexity may be worst than the naïve one, when $min_{r \in R}\{g(BST_i^r, l_i^r)\} = |\Lambda_i|$;
  - But even though such case may not be so frequent, and besides that choosing the threshold $min_{r \in R}\{g(BST_i^r, l_i^r)\} \leqslant |\Lambda_i|$ can be a good choice, since $|\Lambda_i|$ grows exponentially.

Algorithm

Code

Algorithm 1 Labeling algorithm

Require: $D(V, A), c_a \quad \forall a \in A, d_a^r \quad \forall a \in A, r \in R$

$\Lambda_s \leftarrow \{(\{0\}^{|R|} \cup (1,1) \cup \{0\}^{|V|-1}, 0)\}$;
$\Lambda_i \leftarrow \emptyset \quad \forall i \in V \backslash \{s\}$;
$N \leftarrow \{s\}$;
while $N \neq \emptyset$ **do**
    Let $i \in N$ be a N arbitrary node;
    for all $(i,j) \in \delta^+(i)$ **do**
        $F_{ij} \leftarrow \emptyset$;
        for all $S_i \in \Lambda_i$ **do**
            if $f(S_i, j)$ **then**
                $F_{ij} \leftarrow F_{ij} \cup \{E(S_i, j)\}$;
            end if
        end for
        $\Lambda_j \leftarrow N_D(F_{ij} \cup \Lambda_j)$;
        if Λ_j has changed **then**
            $N \leftarrow N \cup \{j\}$;
        end if
    end for
    $N \leftarrow N \backslash \{i\}$;
end while
return $\min_{(L_t, C_t) \in \Lambda_t} \{C_t\}$ if $\Lambda_t \neq \emptyset$ else ∞;

Algorithm

Example

State before line 5:

- $\Lambda_s = \{((0,0,1,1,0,0,0),0)\}$;
- $\Lambda_2 = \Lambda_3 = \Lambda_t = \emptyset$;
- $N = \{\{s\}\}$;

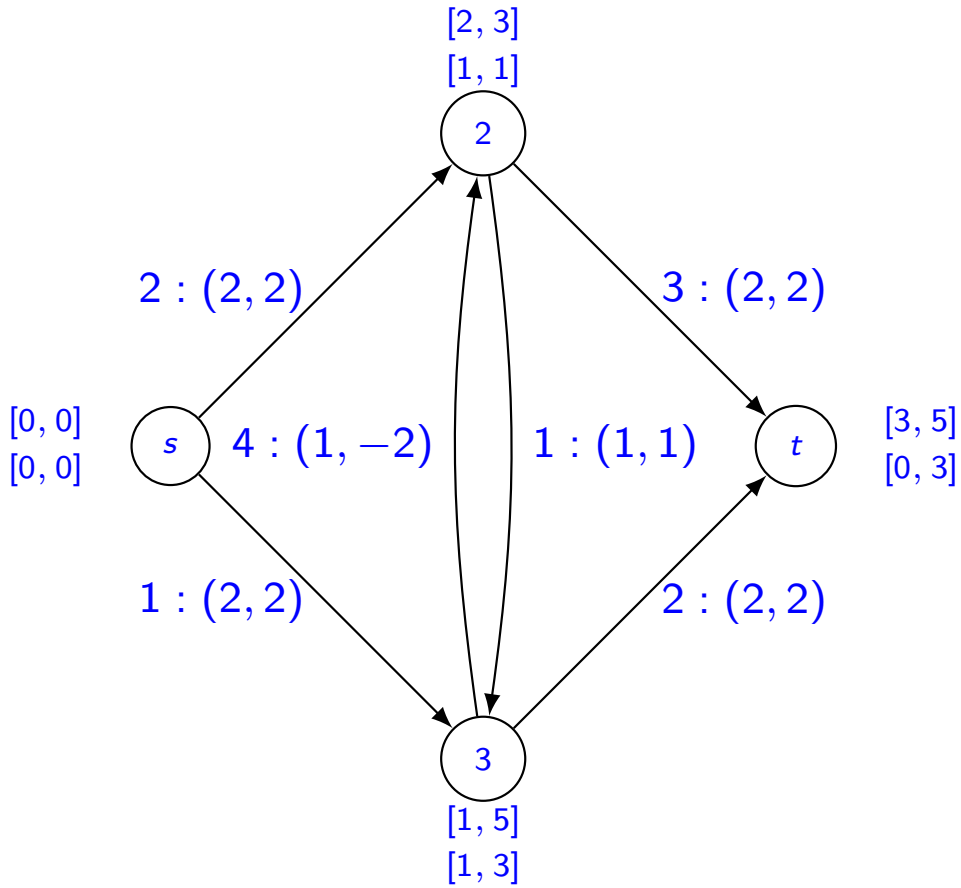

Figure: A ERCSPP instance digraph example.

Algorithm

Example

State at the end of 1st iteration of the while at line 5:

- $\Lambda_s = \{((0,0,1,1,0,0,0),0)\}$;
- $\Lambda_2 = \emptyset$;
- $\Lambda_3 = \{((2,2,2,1,0,1,0),1)\}$;
- $\Lambda_t = \emptyset$;
- $N = \{\{3\}\}$;

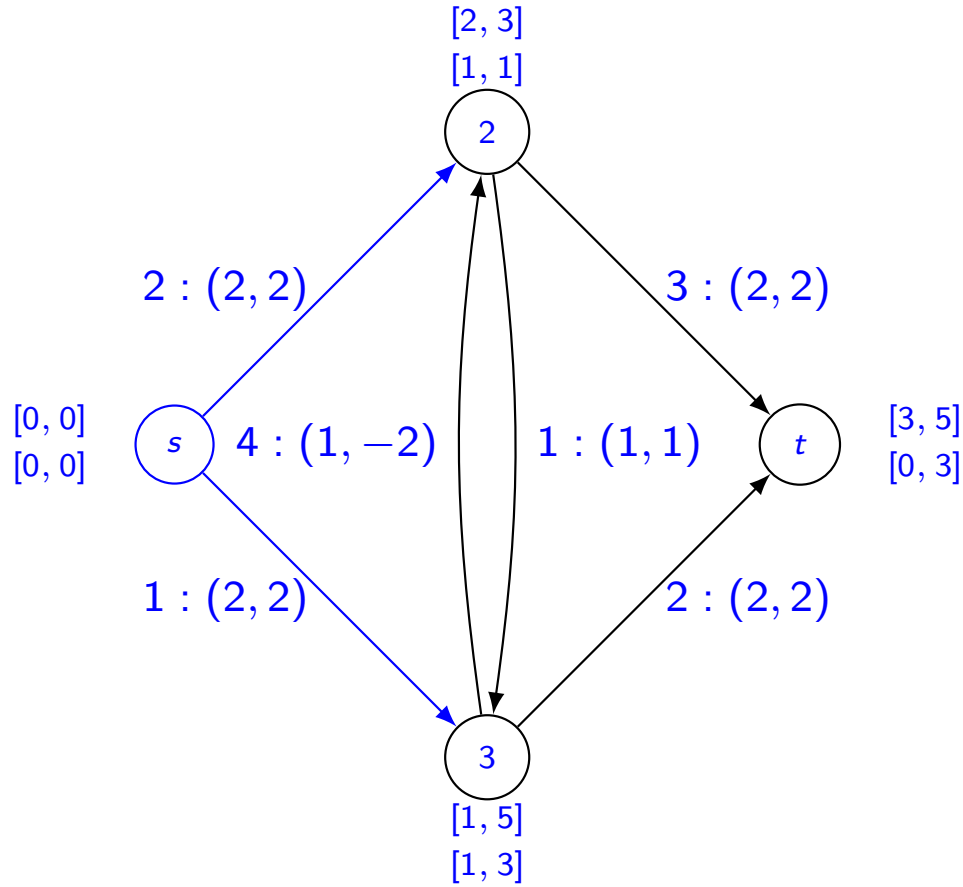

Figure: Arcs explored the for loop at line 5 when $i = s$.

Algorithm

Example

State at the end of 2rd iteration of the while at line 5:

- $\Lambda_s = \{((0,0,1,1,0,0,0),0)\}$;
- $\Lambda_2 = \{((3,0,3,1,1,1,0),5)\}$;
- $\Lambda_3 = \{((2,2,2,1,0,1,0),1)\}$;
- $\Lambda_t = \emptyset$;
- $N = \{\{2\}\}$;

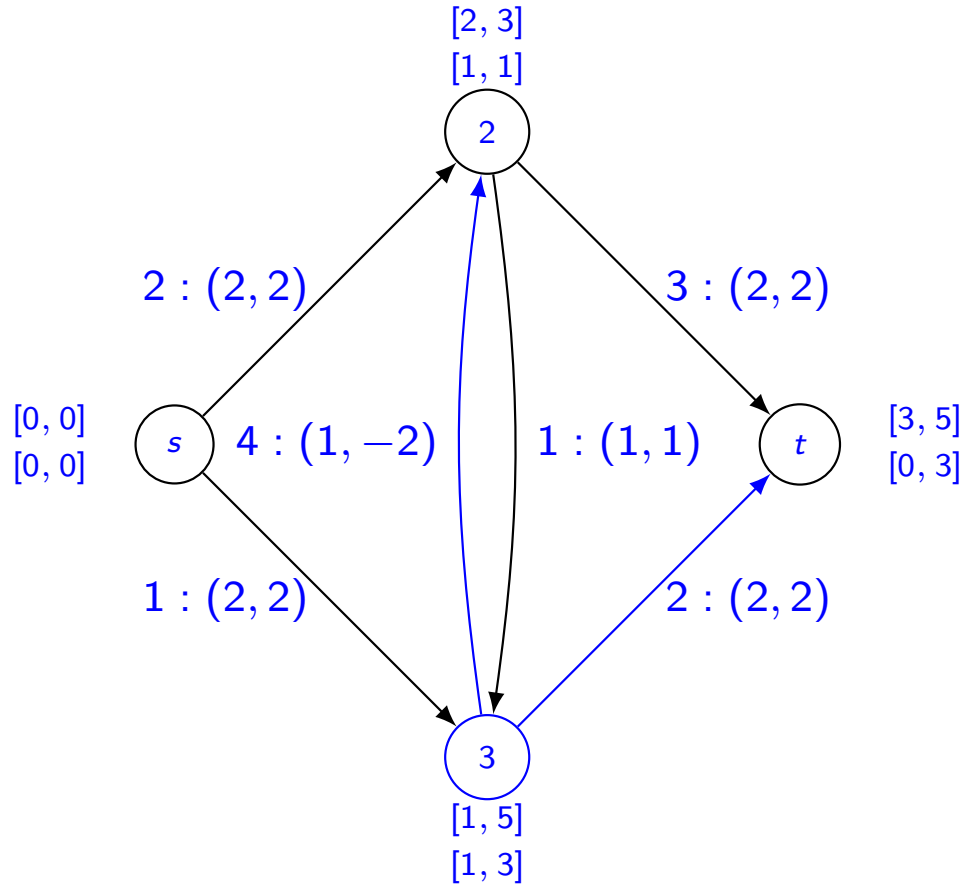

Figure: Arcs explored the for loop at line 5 when $i = 3$.

Algorithm

Example

State at the end of 3nd iteration of the while at line 5:

- $\Lambda_s = \{((0,0,1,1,0,0,0),0)\}$;
- $\Lambda_2 = \{((3,0,3,1,1,1,0),5)\}$;
- $\Lambda_3 = \{((2,2,2,1,0,1,0),1)\}$;
- $\Lambda_t = \{((5,2,4,1,1,1,1),8)\}$;
- $N = \{\{t\}\}$;

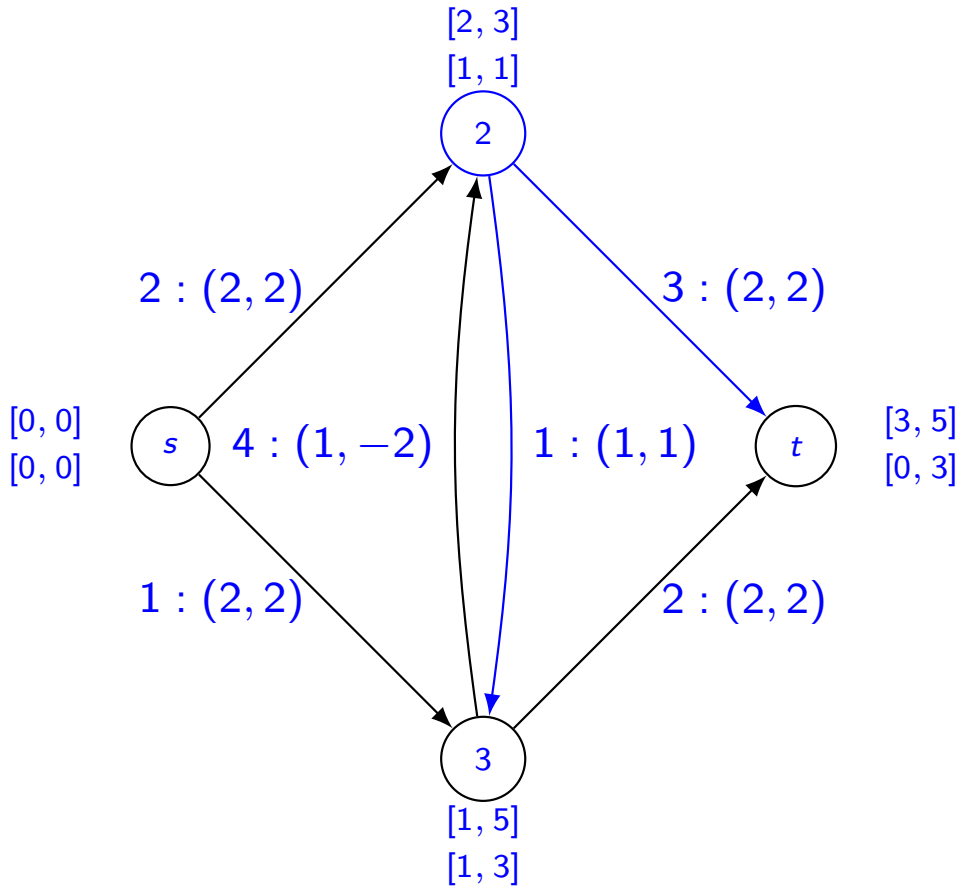

Figure: Arcs explored the for loop at line 5 when $i = 2$.

Algorithm

Example

State at the end of 4th iteration of the while at line 5:

- $\Lambda_s = \{((0,0,1,1,0,0,0),0)\}$;
- $\Lambda_2 = \{((3,0,3,1,1,1,0),5)\}$;
- $\Lambda_3 = \{((2,2,2,1,0,1,0),1)\}$;
- $\Lambda_t = \{((5,2,4,1,1,1,1),8)\}$;
- $N = \emptyset$;

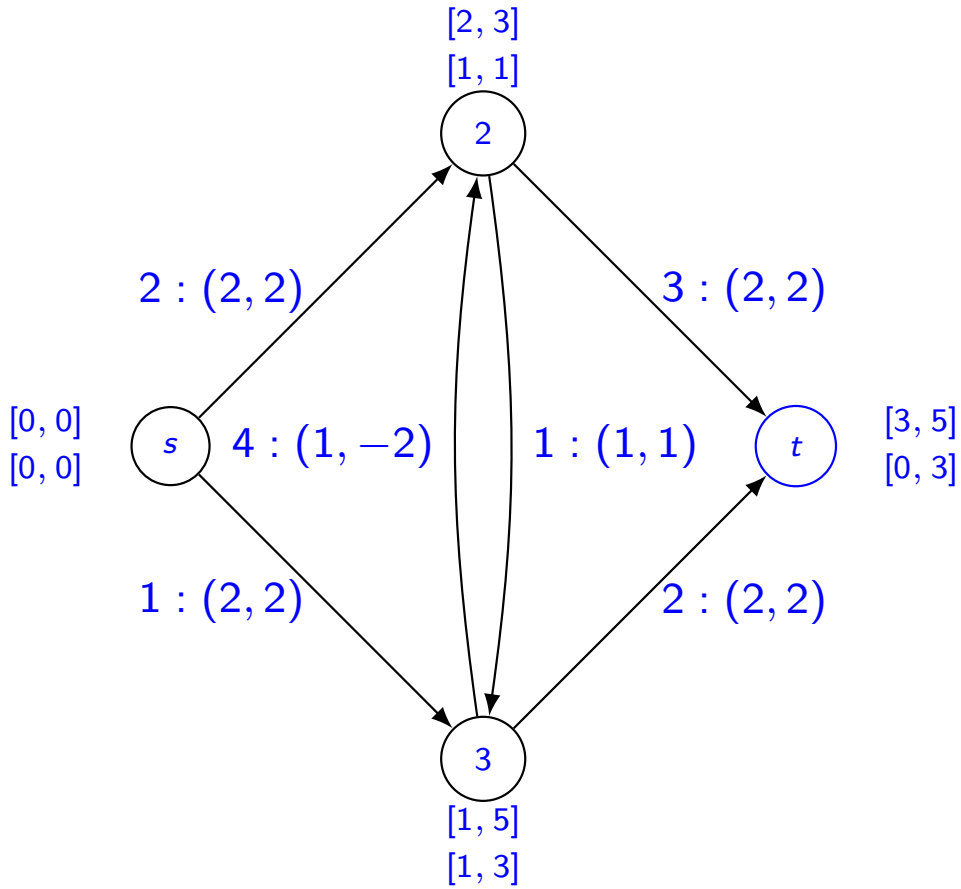

Figure: Arcs explored the for loop at line 5 when $i = t$.

Bi-directional algorithm[4]

[4] Giovanni Righini and Matteo Salani. "Symmetry helps: Bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints". In: *Discrete Optimization* 3.3 (2006). Graphs and Combinatorial Optimization, pp. 255–273. ISSN: 1572-5286. DOI: https://doi.org/10.1016/j.disopt.2006.05.007. URL: https://www.sciencedirect.com/science/article/pii/S1572528606000417.

Bi-directional search

Previous algorithm analysis

- The number of generated states increases rapidly according to the size of the instance;
- Among the reasons it is that any node i label extension generates as many other labels as the number of i possible successors;
- States are pruned only when they are dominated;

Bi-directional search

Improvements

Consider labels being propagated both from s to t and vice-versa. Let

- P'_j be a feasible j-t-path in D;
- Λ_i^f and Λ_i^b be the sets of forward and backward labels of node $i \in V$;
- $E^b(S_j, i) = (\max\{l_j^r - d_{ij}^r, (max_{i' \in V} e_i^r) - b_i^r\} : r \in R) \cup (s_i + 1, v_i^1, ..., v_i^j + 1, ..., v_i^{|V|})$ be the function that returns the label resulting from the backwards extension of a path P'_j by a node i ($O(|R|)$).
- $$f^b(S_j, i) = \begin{cases} \text{true}, & \text{if } \forall r \in R(l_j^r - d_{ij}^r \leqslant e_i^r) \wedge \neg v_j^i \\ \text{false}, & \text{otherwise} \end{cases}$$

  be a function that says whether a backward extension is feasible ($O(|R|)$);

Bi-directional search

Improvements

Let

- $i, j \in V : i \neq j$ be two distinct nodes;
- $S_i = (L_i, C_i) \in \Lambda_i^f$ be a forward state of node i; and
- $S_j = (L_j, C_j) \in \Lambda_j^b$ be a backward state of node j.

Definition

Forward state S_i and a backward state S_j can be feasibly joined through arc (i, j) if $(\neg((v_i^1, ..., v_i^{|V|}) \vee (v_j^1, ..., v_j^{|V|}))$, and $l_j^r - d_{ij}^r \geqslant l_i^r$.

A path from s to t is detected each time a forward state in Λ_i^f and a backward state in Λ_j^b can be feasibly joined through arc (i, j).

Bi-directional search

Algorithm: Forward extension procedure

Algorithm 2 Forward extension algorithm

1: Function{Forward extension}$\{i \in V, \Lambda_i^f, N\}$
2: **for all** $(i,j) \in \delta^+(i)$ **do**
3: $F_{ij} \leftarrow \emptyset$;
4: **for all** $S_i \in \Lambda_i^f$ **do**
5: **if** $f(S_i, j)$ **then**
6: $F_{ij} \leftarrow F_{ij} \cup \{E(S_i, j)\}$;
7: **end if**
8: **end for**
9: $\Lambda_j^f \leftarrow N_D(F_{ij} \cup \Lambda_j^f)$;
10: **if** Λ_j^f has changed **then**
11: $N \leftarrow N \cup \{j\}$;
12: **end if**
13: **end for**
14: $N \leftarrow N \backslash \{i\}$;
15: EndFunction

Bi-directional search

Algorithm: Backward extension procedure

Algorithm 3 Forward extension algorithm

1: Function{Backward extension}$\{i \in V, \Lambda_i^b, N\}$
2: **for all** $(j,i) \in \delta^-(i)$ **do**
3: $F_{ji} \leftarrow \emptyset$;
4: **for all** $S_i \in \Lambda_i^b$ **do**
5: **if** $f^b(S_i, j)$ **then**
6: $F_{ji} \leftarrow F_{ji} \cup \{E^b(S_i, j)\}$;
7: **end if**
8: **end for**
9: $\Lambda_j^b \leftarrow N_D(F_{ji} \cup \Lambda_j^b)$;
10: **if** Λ_j^b has changed **then**
11: $N \leftarrow N \cup \{j\}$;
12: **end if**
13: **end for**
14: $N \leftarrow N \backslash \{i\}$;
15: EndFunction

Bi-directional search

Algorithm: Search

Algorithm 4 Search

Require: $D(V, A), c_a \quad \forall a \in A, d_a^r \quad \forall a \in A, r \in R$

$\Lambda_s^f \leftarrow \{(\{0\}^{|R|} \cup (1,1) \cup \{0\}^{|V|-1}, 0)\}$;
$\Lambda_t^b \leftarrow \{(e_t^r : r \in R) \cup (1,1) \cup \{0\}^{|V|-1}, 0)\}$;
$\Lambda_i^f \leftarrow \emptyset \quad \forall i \in V \backslash \{s\}$;
$\Lambda_i^b \leftarrow \emptyset \quad \forall i \in V \backslash \{t\}$;
$N \leftarrow \{s, t\}$;
while $N \neq \emptyset$ **do**
    Let $i \in N$ be a N arbitrary node;
    Forward extension$\{i, \Lambda_i^f, N\}$;
    Backward extension$\{i, \Lambda_i^b, N\}$;
    $N \leftarrow N \backslash \{i\}$;
end while
{Merge}$\{\Lambda^f, \Lambda^b, N\}$;

Bi-directional search

Strategies

- Bounding for pruning;
- Halfway bounding for the number of extensions;
- Bounding for monotone resources;
- Solutions uniqueness.

Bi-directional search

Bounding for pruning

Let

- $S_i = (L_i, C_i) \in \Lambda_i^f$ be a forward state of node $i \in V$;
- $\mathcal{S} = \{j \in V : v_i^j\} = V(P_i)$ be the set of nodes in P_i;
- $m_r^f(S_i, j) = min_{k \notin \mathcal{S} \setminus \{i\}}\{d_{kj}^r\}$ be a lower bound to the consumption of resource $r \in R$ when a forward state S_i is extended to $j \in V$;
- $S_i^{'} = (L_i^{'}, C_j^{'}) \in \Lambda_i^b$ be a forward state of node $i \in V$;
- $m_r^b(S_i, j) = min_{k \notin \mathcal{S} \setminus \{i\}}\{d_{jk}^r\}$ be a lower bound to the consumption of resource $r \in R$ when the backward state $S_i^{'}$ is extended to $j \in V$;

Bi-directional search

Bounding for pruning

$$min \quad \sum_{j \in V \setminus S} \lambda_j min_{k \notin S \setminus \{i\}} \{c_{kj}\} \tag{1}$$

$$s.t. \quad l_i^r + \sum_{j \in V \setminus S} \lambda_j m_r^f(S_i, j) \leqslant e_r^i \qquad \forall r \in R \tag{2}$$

$$\lambda_j \in \mathbb{B} \qquad \forall j \in V \setminus S \tag{3}$$

Let $\bar{C}_i$ be the optimal solution of the above knapsack formulation for the label S_i, and UB be an upper bound on the ERCSPP solution. If $C_i - \bar{C}_i \geqslant UB$ then the state S_i can be chopped off.

Bi-directional search

Halfway bounding for the number of extensions

$$min \qquad \sum_{j \in V \setminus S} \lambda_j \tag{4}$$

$$s.t. \qquad I_i^r + \sum_{j \in V \setminus S} \lambda_j m_r^f(S_i, j) \leqslant e_r^i \qquad \forall r \in R \tag{5}$$

$$\lambda_j \in \mathbb{B} \qquad \forall j \in V \setminus S \tag{6}$$

If the optimal solution p of the above knapsack formulation is $\leqslant |S|$ then the state S_i has reached its halfway, since we are working with bi-directional search, then we can stop the extensions of this state.

Bi-directional search

Bounding for monotone resources

Let $r \in R$ be a resource, r is said to be monotone iff $d_a^r \geqslant 0$ $\forall a \in A$. If $l_i^r \geqslant \frac{e_t^r}{2}$ then the state S_i has reached its resource halfway, then we can stop the extensions of this state.

Bi-directional search

Solutions uniqueness

- Another issue to be considered comes from the need of generating many different columns with negative reduced cost when using the ERCSPP as a pricing subproblem in a branch-and-price;
- The bounded bi-directional dynamic programming algorithm can provide duplicate solutions:
  - Consider for instance an s-t path including the sequence (i, j, k);
  - Eventually, the forward states for vertices i and j and backward states for vertices j and k are generated;
  - Hence the same solution is obtainable by merging a forward state of i with a backward state of j, or merging a forward state of j with a backward state of k;
  - If we need to store in some data structure all columns with negative reduced cost, the duplicate columns cannot be discarded on the basis of their cost and their identification may be computationally expensive.

Bi-directional search

Solutions uniqueness

- Then, we accept an s-t path only when it is produced by the join of a forward state and a backward state, which the forward and backward consumptions of a monotone resource are as close as possible to half the overall consumption ($\frac{e_t^r}{2}$), that is the two states are as close as possible to the halfway point along the s-t path;
- Let ρ_f and ρ_b be a monotone resource consumption in forward and backward paths;
- Among all possible pairs of forward and backward states producing the same s-t path we choose the one which $\psi = |\rho_f - \rho_b|$ is minimum;
- In the case of a tie, i.e., two available mergings where ψ is minimum, we choose the one with $\rho_f > \rho_b$;
- This test guarantees that each s–t path is generated only once.

Bi-directional search

Algorithm: Merge

Let

- $$f(S_i, S_j) = \begin{cases} \text{true}, & \text{if } \forall r \in R(l_j^r - d_{ij}^r \leqslant e_i^r) \wedge \\ & (v_i^1, ..., v_i^{|V|}) \implies \neg(v_j^1, ..., v_j^{|V|}) \\ \text{false}, & \text{otherwise} \end{cases}$$

  be a function that says whether a merge between a forward label S_i and a backward label S_j is feasible;
- $H(S_i, S_j)$ be the function that checks if the s-t path obtained from merging the forward label S_i and the backward label S_j satisfies the halfway conditions outlined before (Bounding for monotone resources, and Halfway bounding for the number of extensions);
- $Save(S_i, S_j)$ a function tha saves the solution obtained from the two merging of S_i and S_j.

Bi-directional search

Algorithm: Merge

Algorithm 5 Merge forward and backward paths algorithm

1: Function{Merge forward and backward paths}$\{i, \Lambda_i^f, \Lambda_i^b\}$
2: Let $\phi_i^f = min_{(L_i,C_i)\in\Lambda_i^f}\{C_i\}$ be the minimum cost among all labels in Λ_i^f;
3: Let $\phi_i^b = min_{(L_i,C_i)\in\Lambda_i^b}\{C_i\}$ be the minimum cost among all labels in Λ_i^b;
4: Let $\phi^b = min_{i\in V}\{\phi_i^b\}$ be the minimum cost among all labels in Λ^b;
5: **for all** $i \in V$ **do**
6: **if** $\phi_i^f + min_{j\in V\setminus\{i\}} c_{ij} + \phi^b < UB$ **then**
7: **for all** $S_i = (L_i, C_i) \in \Lambda_i^f$ **do**
8: **if** $C_i + min_{j\in V\setminus\{i\}} c_{ij} + \phi^b < UB$ **then**
9: **for all** $j \in V$ **do**
10: **if** $C_i + c_{ij} + \phi_j^b < UB$ **then**
11: **for all** $S_j = (L_j, C_j) \in \Lambda_j^b$ **do**
12: **if** $C_i + c_{ij} + C_j < UB \wedge f(S_i, S_j) \wedge H(S_i, S_j)$ **then**
13: $Save(S_i, S_j)$;
14: **end if**
15: **end for**
16: **end if**
17: **end for**
18: **end if**
19: **end for**
20: **end if**
21: **end for**


Dominique Feillet et al. "An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems". In: *Networks* 44.3 (2004), pp. 216–229. DOI: 10.1002/net.20033.


Giovanni Righini and Matteo Salani. "Symmetry helps: Bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints". In: *Discrete Optimization* 3.3 (2006). Graphs and Combinatorial Optimization, pp. 255–273. ISSN: 1572-5286. DOI: https://doi.org/10.1016/j.disopt.2006.05.007. URL: https://www.sciencedirect.com/science/article/pii/S1572528606000417.