

Label-Setting Algorithm for Shortest Path Problems

Bjørn Petersen

DIKU, University of Copenhagen

Recent Research Results

21st November 2006

Outline



- Introduction
- Various relaxations
- Label Setting
- Domination
- ESPPRC
- k -cyc-SPPRC
- Subset penalty
- Bi-directionality
- Test data
- Literature
- Assignments

Introduction

Shortest path problem with resource constraints:

- Weighted directed graph $G(V, E)$
- Nodes V , edges E and resources R
- For each edge $e \in E$ and resource $r \in R$
 - Lower limit $a_r(e)$
 - Upper limit $b_r(e)$
 - Consumption $c_r(e)$
- Path p satisfies the limits for all resources $r \in R$.
- Objective: Find shortest path p from $s \in V$ to $t \in V$

Note: Upper and lower limits and consumptions on the nodes can be “pushed” onto the edges

Mathematical model of ESPPRC

Only one resource considered (e.g. time)

$$\min \sum_{(i,j) \in E} c_{ij} x_{ij} \quad (1)$$

$$\text{s.t.} \quad \sum_{(s,i) \in \delta^+(s)} x_{si} = 1 \quad (2)$$

$$\sum_{(i,t) \in \delta^-(t)} x_{it} = 1 \quad (3)$$

$$\sum_{(i,j) \in \delta^+(i)} x_{ij} \leq 1 \quad \forall i \in V \setminus \{s, t\} \quad (4)$$

$$\sum_{(j,i) \in \delta^-(i)} x_{ji} = \sum_{(i,j) \in \delta^+(i)} x_{ij} \quad \forall i \in V \setminus \{s, t\} \quad (5)$$

$$\sum_{(j,i) \in \delta^-(i)} (T_{ji} + \tau_{ji} x_{ji}) \leq \sum_{(i,j) \in \delta^+(i)} T_{ij} \quad \forall i \in V \setminus \{s, t\} \quad (6)$$

$$a_{ij} x_{ij} \leq T_{ij} \leq b_{ij} x_{ij} \quad \forall (i, j) \in E \quad (7)$$

$$x_{ij} \in \{0, 1\} \quad \forall (i, j) \in E \quad (8)$$

Cost of resource $\bar{c}(e(i, j)) = \tau_{ij}$

Various relaxations

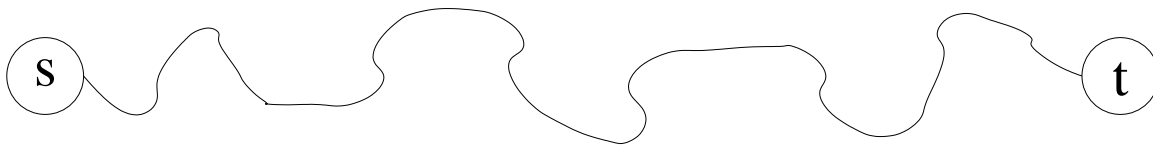
Difficult: Negative cost cycles may appear. Hard to find elementary paths

Make the shortest path problem easier to solve (in e.g. VRP context)

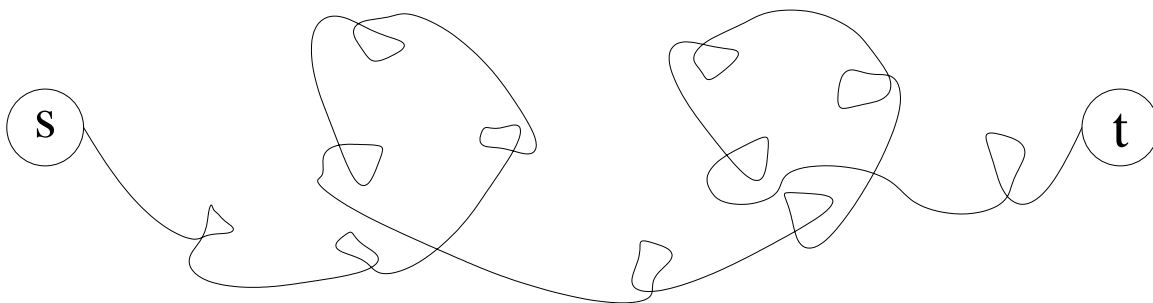
- Allow nodes to be visited multiple times
- Not all routes are feasible due to cycling
 - Cannot be part of an optimal solution
 - Handled in the master problem

The relaxation can be done in several grades:

- Disallow all cycles (elementary)

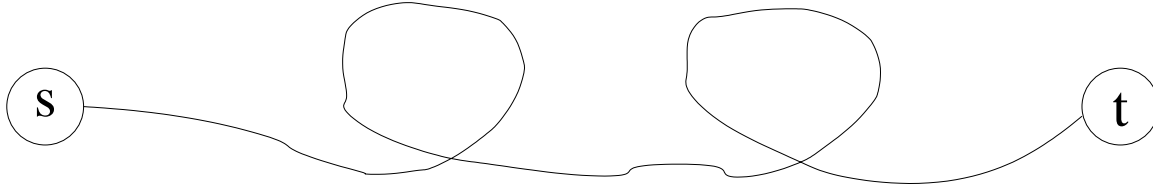


- Allow all cycles

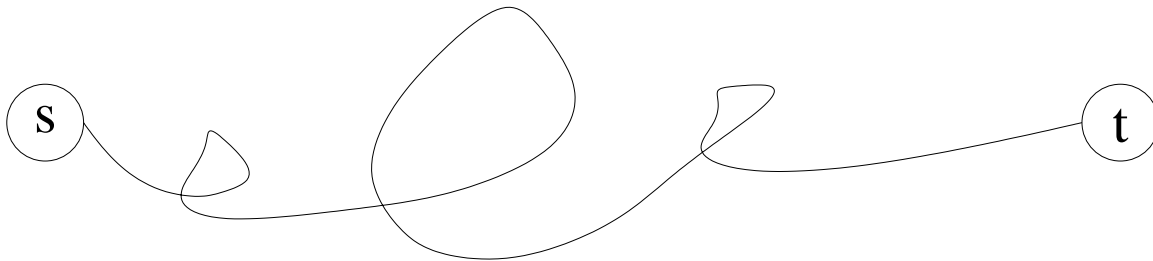


Various relaxations (continued)

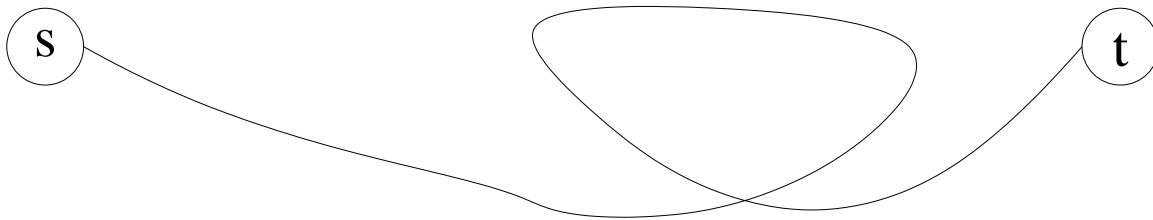
- Disallow small cycles, i.e. smaller than k edges



- Disallow certain customers on all cycles



- Disallow all small cycles and certain customers on all cycles



Various relaxations (continued)

Disallowing small cycles ($size \leq k$), k -cyc-SPPRC by Irnich and Villeneuve (2006)

Disallow certain customers on all cycles, say all in $S \subseteq V$
Partial ESPPRC, suggested by Dumitrescu and Boland (2005)

Disallowing all cycles ($S = V$) ESPPRC, Beasley and Christofides (1989), Chabrier (2005), Danna and Pape (2005), Dumitrescu (2002), Feillet et al. (2004)

Combining k -cyc-SPPRC and Partial ESPPRC is new

Nodes in S modeled as resources (node resources)

k -cyc-SPPRC taken care of explicitly (Hole Sets)

Note: if for $u, v \in V$ no feasible path P with $u \in P$ and $v \in P$ can exist then u and v can share resource bit

Label setting

Concepts of label setting

- Find shortest path from node s to node t
- Graph $G(V, E)$ may contain cycles of negative cost
- Create all possible paths
- Equivalent to dynamic programming with sparse representation

Labels at node v represent partial paths from s to v .

Attributes for a label L :

$\bar{v}(L)$ The node L belongs to

$p(L)$ The parent of L (the label extended to make L)

$r(L)$ The accumulated consumption of resource $r \in R$

$\pi(L)$ Ordered set of last $k - 1$ visited nodes (k -cyc-SPPRC)

Label setting (continued)

Assumptions:

$\forall r \in R$ strongly bounded from above

$\forall r \in R$ weakly bounded from below (wait for lower bound)

Extension function c is non-decreasing, i.e.,

$$x \leq y \Rightarrow c(x) \leq c(y) \quad \forall x, y$$

At least one resource r_m has a monotone cost function

I.e. L_u extended by edge $e = (u, v)$ to become L_v must satisfy

$$r(L_u) \leq b_r(e), \quad \forall r \in R \quad (9)$$

$$r(L_v) = \max\{r(L_u) + c_r(e), a_r(e)\}, \quad \forall r \in R \quad (10)$$

$$v \notin w, \quad \forall w \in \pi(L_u) \quad (11)$$

Label setting (continued)

Find shortest path by complete enumeration (alternatively by filling a dynamic programming table)

- 1 Make the first label
- 2 Find untreated labels with smallest accumulated consumption of r_m and extend it to make new labels
- 3 If more untreated labels exist go to step 2 else stop

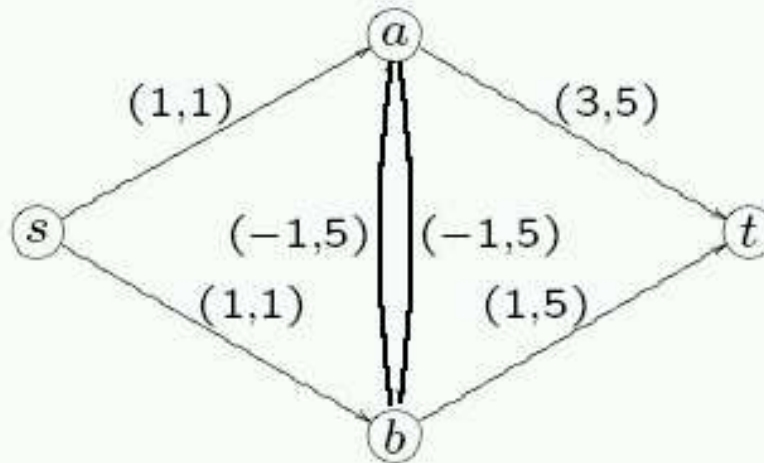
ENUM-LABEL-SETTING(G, s, t)

```
1   $L_{init} \leftarrow \text{FIRST-LABEL}(s)$ 
2   $PQ.\text{ENQUEUE}(L_{init})$ 
3  while  $PQ \neq \emptyset$ 
4      do  $L \leftarrow PQ.\text{DEQUEUE}()$ 
5          for each node  $v \in \text{EXTENDABLES}(L)$ 
6              do  $L_v \leftarrow \text{EXTEND-LABEL}(L, v)$ 
7                  if  $\bar{v}(L_v) = t$ 
8                      then  $\text{STORE-SOLUTION}(L_v, sol)$ 
9                  if  $\bar{v}(L_v) \neq t$ 
10                     then  $PQ.\text{ENQUEUE}(L_v)$ 
11 return  $sol$ 
```

Procedure $\text{Extend-Label}(L, v)$ creates new label L_v with $\bar{v}(L_v) = v, p(L_v) = L, r(L_v) = r(L) + c_r(v(L), v) \quad \forall r \in R$ and $\pi(L_v)$ updated according to k

Extendables returns the nodes that can be extended to

Label setting (example)



Graph of an instance of the shortest path problem with a time constraint. The path has to consume less than 20 units of time. Notation on the graph (*cost, time*).

Iteration	Untreated	Extends to	New labels
<i>I</i>	L_1	$L_1 \rightsquigarrow a, b$	$L_2 = \{a, 1, 1\}$ $L_3 = \{b, 1, 1\}$
<i>II</i>	L_2, L_3	$L_2 \rightsquigarrow b, t$	$L_4 = \{b, 0, 6\}$ $L_5 = \{t, 4, 6\}$
<i>III</i>	L_3, L_4	$L_3 \rightsquigarrow a, t$	$L_6 = \{a, 0, 6\}$ $L_7 = \{t, 2, 6\}$
<i>IV</i>	L_4, L_6	$L_4 \rightsquigarrow a, t$	$L_8 = \{a, -1, 11\}$ $L_9 = \{t, 1, 11\}$
<i>V</i>	L_6, L_8	$L_6 \rightsquigarrow b, t$	$L_{10} = \{b, -1, 11\}$ $L_{11} = \{t, 3, 11\}$
<i>VI</i>	L_8, L_{10}	$L_8 \rightsquigarrow b, t$	$L_{12} = \{b, -2, 16\}$ $L_{13} = \{t, 2, 16\}$
<i>VII</i>	L_{10}, L_{12}	$L_{10} \rightsquigarrow a, t$	$L_{14} = \{a, -2, 16\}$ $L_{15} = \{t, 0, 16\}$
<i>VIII</i>	L_{12}, L_{14}	L_{12}	
<i>IX</i>	L_{14}	L_{14}	

Domination

Exponential number of labels

All labels not necessarily in optimal solution, so some may not be necessary to consider

Reduce running time by fathoming labels not leading to unique optimal solution

Definition 1. The set of all feasible paths from label L to node u considering the resource consumption of label L is defined as $\mathcal{F}(L, u)$

Definition 2. The set of all feasible paths from label L to node u considering the cycle restrictions (k -cycle elimination and the partial elementarity) is defined to be $\mathcal{S}(L, u)$

Definition 3. All feasible extensions of label L is defined as:

$$\mathcal{E}(L) = \mathcal{F}(L, t) \cap \mathcal{S}(L, t)$$

Domination (continued)

Definition 4. A set of labels \mathcal{L}_i dominates label L_j if:

$$\bar{v}(L_i) = \bar{v}(L_j) \quad \forall L_i \in \mathcal{L}_i \quad (12)$$

$$\bar{c}(L_i) \leq \bar{c}(L_j) \quad \forall L_i \in \mathcal{L}_i \quad (13)$$

$$\mathcal{E}(L_j) \subseteq \bigcup_{L_i \in \mathcal{L}_i} \mathcal{E}(L_i) \quad (14)$$

$$\forall \varepsilon \in \mathcal{E}(L_j) \quad \exists L_i \in \mathcal{L}_i : \varepsilon \in \mathcal{E}(L_i) \wedge \bar{c}(L_i) \leq \bar{c}(L_j)$$

In other words; if L_j is dominated then it is not part of an unique optimal solution

Let $\mathcal{P}(L) = P(L) \cup \varepsilon : \forall \varepsilon \in \mathcal{E}(L)$ be the set of paths that includes label L

Proposition 5. For any path $p \in \mathcal{P}(L)$ if $\mathcal{L} \prec_{dom} L$ then there exist an alternative path p' where $c(p') \leq c(p)$

Proof. By Definition 4 there exists \mathcal{L}' such that:

$$\mathcal{E}(L) \subseteq \mathcal{E}(\mathcal{L}')$$

Thus for any $\varepsilon \in \mathcal{E}(L)$ there exists some label $L_i \in \mathcal{L}'$ where $\varepsilon \in \mathcal{E}(L_i)$. For any path $p \in \mathcal{P}(L)$ where $p = P(L) \cup \varepsilon$ it follows that there exist some $p' = P(L_i) \cup \varepsilon : L_i \in \mathcal{L}' \wedge \varepsilon \in \mathcal{E}(L_i)$ and since $c(L_i) \leq c(L)$ it follows that $c(p') \leq c(p)$ \square

Domination (continued)

To determine if (14) holds can be cumbersome

Sufficient criteria for (14) that can be calculated faster is sought

If $r(L_i) \leq r(L_j) \quad \forall r \in R$ then no resources are limiting extensions of L_i compared to L_j

Elementary SPPRC

Proposition 6 (Sufficient condition). *For ESPPRC a label L_i dominates label L_j if:*

$$\bar{v}(L_i) = \bar{v}(L_j) \quad (15)$$

$$r(L_i) \leq r(L_j) \quad \forall r \in R \quad (16)$$

Condition (14) is satisfied with (16) since elementary requirement modeled with resources

Domination (continued)

Feillet et al. (2004) suggested eager update of node resources instead of lazy, i.e. marking nodes when not reachable instead of visited

Definition 7. Given label L with $\bar{v}(L) = u$ then node $v \in V$ is *unreachable* if v is visited by L or if a resource window is violated, e.g.:

$$\exists r \in R \quad r(L) + l_r(u, v) > b_r(v)$$

where $l_r(u, v)$ is a lower bound on the consumption of resource r on all feasible paths from u to v .

Domination (continued)

Only brief description given for k -cyc-SPPRC

Cycles of constant size k can be avoided without increasing complexity of algorithm (no more than a constant factor)

NOTE: Constant factor is large and grows with:

- Running time: $O(kk!^3)$
- Space: $O(k!^2)$

Intuitively simple – algorithmically complicated

$$\begin{array}{l}
 L_w^1: \quad u \quad \rightarrow \quad v \quad \rightarrow \quad w \quad \rightarrow \quad \bar{u}/\bar{v} \quad \rightarrow \quad \bar{v} \\
 L_w^2: \quad \bar{u} \quad \rightarrow \quad \bar{u}/\bar{v} \quad \rightarrow \quad w \quad \rightarrow \quad u \quad \rightarrow \quad v \\
 L_w^3: \quad \bar{u} \quad \rightarrow \quad \bar{u} \quad \rightarrow \quad w \quad \rightarrow \quad u \quad \rightarrow \quad \bar{u} \\
 L_w^4: \quad \bar{v} \quad \rightarrow \quad \bar{u}/\bar{v} \quad \rightarrow \quad w \quad \rightarrow \quad v \quad \rightarrow \quad u \\
 L_w^5: \quad \bar{v} \quad \rightarrow \quad \bar{v} \quad \rightarrow \quad w \quad \rightarrow \quad v \quad \rightarrow \quad \bar{v} \\
 L_w^6: \quad \bullet \quad \rightarrow \quad \bar{u} \quad \rightarrow \quad w \quad \rightarrow \quad \bar{u} \quad \rightarrow \quad u(*) \\
 L_w^7: \quad \bullet \quad \rightarrow \quad \bar{v} \quad \rightarrow \quad w \quad \rightarrow \quad \bar{v} \quad \rightarrow \quad v
 \end{array}$$

Domination (continued)

Algorithm works by set operators on implicit set of extensions:

\cap Intersection

\subseteq Subset of

Set of extensions of label L only considering k -cyc-SPPRC defined by $\overline{\mathcal{E}}(L)$

Hole sets $H(L)$ is the set complement to $\overline{\mathcal{E}}(L)$

Using De Morgans rule:

$$\overline{\mathcal{E}}(L) \subseteq \bigcup_{L_i \in \mathcal{L}} \overline{\mathcal{E}}(L_i) \Leftrightarrow \bigcap_{L_i \in \mathcal{L}} H(L) \subseteq H(L_i)$$

Domination (continued)

Combining k -cyc-SPPRC and Partial ESPPRC

Relaxed dominance criteria:

Proposition 8 (Sufficient condition). *A set of labels \mathcal{L}_i dominates label L_j if:*

$$\bar{v}(L_i) = \bar{v}(L_j) \quad \forall L_i \in \mathcal{L}_i \quad (17)$$

$$r(L_i) \leq r(L_j) \quad \forall r \in R, \forall L_i \in \mathcal{L}_i \quad (18)$$

$$\overline{\mathcal{E}}(L_j) \subseteq \bigcup_{L_i \in \mathcal{L}_i} \overline{\mathcal{E}}(L_i) \quad (19)$$

and node resources are set according to Definition 7.

The \subseteq -operator and \bigcup -operator on ordered sets defined as by Irnich and Villeneuve (Hole Sets)

Note: π only defined on $v \notin S$ to restrict solution space

Note: Condition (18) can be tightened by being lazy with $r(L_i)$ and eager with $r(L_j)$

Note: If $k \leq 1$ Condition (19) is automatically fulfilled, so $|\mathcal{L}_i| = 1$

Subset penalty

Introduced by (Jepsen et. al 2006)

Given $T \subseteq V$, $|T| = n$, $0 < k \leq n$ and $\sigma \geq 0$

A penalty σ must be payed for each k visits to the nodes in T .

Apology: k here is unrelated to the k in k -cyc-SPPRC

Let $E(L)$ be the edges and $V(L)$ the nodes visited on the partial path of label L . Cost of L :

$$\bar{c}(L) = \sum_{e \in E(L)} c_e + \sigma \left\lfloor \frac{|T \cap V(L)|}{k} \right\rfloor$$

Previous dominance criteria of Proposition 8 invalidated since penalty depending on both visited and unvisited nodes

I.e. if label L_i dominates label L_j regarding Proposition 8 then $\bar{c}(L_j) < \bar{c}(L_i)$ could be true later due to penalties payed by L_i but not by L_j

Subset penalty (continued)

It suffices to consider if label L_i is closer to paying a penalty than label L_j

Let $\mathcal{T}(L) = |T \cap V(L)| \bmod k$ be the number of visits made to T since the last penalty was payed

Cost dominance criteria for a single subset penalty:

Proposition 9. *If $\mathcal{T}(L_i) \leq \mathcal{T}(L_j)$ label L_i may dominate label L_j disregarding any additional penalties.*

since:

$$\left\lfloor \frac{|T \cap \mathcal{E}(L_j)| + \mathcal{T}(L_i)}{k} \right\rfloor \sigma \leq \left\lfloor \frac{|T \cap \mathcal{E}(L_j)| + \mathcal{T}(L_j)}{k} \right\rfloor \sigma$$

Proposition 10. *If $\mathcal{T}(L_i) > \mathcal{T}(L_j)$ label L_i may dominate label L_j if a temporary penalty σ is added to the cost of L_i .*

since: $0 \leq \mathcal{T}(L_j) < \mathcal{T}(L_i) \leq k - 1$ applying an additional penalty to L_i gives:

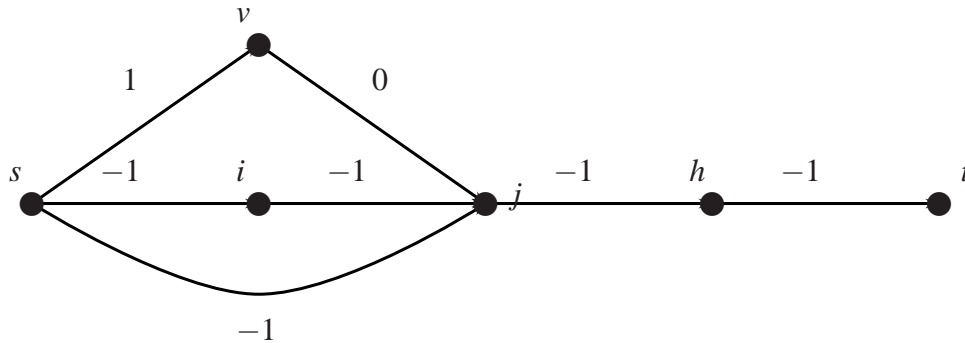
$$\sigma + \left\lfloor \frac{|T \cap \mathcal{E}(L_j)| + \mathcal{T}(L_i)}{k} \right\rfloor \sigma \leq \left\lfloor \frac{|T \cap \mathcal{E}(L_j)| + \mathcal{T}(L_j)}{k} \right\rfloor \sigma$$

Note: if $\mathcal{T}(L_i) + |T \cap \mathcal{E}(L_j)| < k$ it is not possible for ε to visit T enough times to trigger penalty

Subset penalty (continued)

Example:

$$T = \{i, j, h\}, n = 3, k = 2, \sigma = 2$$



L	$\bar{v}(L)$	$\bar{c}(L)$	$V(L)$	$\mathcal{T}_S(L)$
L_1	j	0	$\{s, i, j\}$	0
L_2	j	-1	$\{s, j\}$	1
L_3	j	1	$\{s, v, j\}$	1

Subset penalty (continued)

The new dominance criteria is as follows:

Proposition 11 (Sufficient condition). *Let Q be the set of all subset penalties where*

$$\sigma > 0 \wedge \mathcal{T}(L_i) > \mathcal{T}(L_j)$$

A set of labels \mathcal{L}_i dominates label L_j if:

$$\bar{v}(L_i) = \bar{v}(L_j) \quad \forall L_i \in \mathcal{L}_i \quad (20)$$

$$\bar{c}(L_i) + \sum_{q \in Q} \sigma_q \leq \bar{c}(L_j) \quad \forall L_i \in \mathcal{L}_i \quad (21)$$

$$r(L_i) \leq r(L_j) \quad \forall r \in R \setminus \{\bar{c}\}, \forall L_i \in \mathcal{L}_i \quad (22)$$

$$\overline{\mathcal{E}}(L_j) \subseteq \bigcup_{L_i \in \mathcal{L}_i} \overline{\mathcal{E}}(L_i) \quad (23)$$

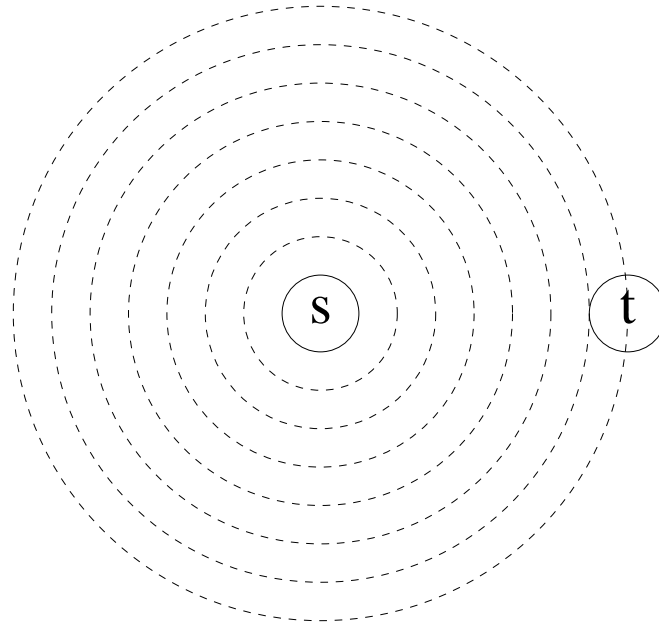
and node resources are set according to Definition 7.

Bi-directionality

Problem: Find the shortest path from node s to node t

Using ideas from Dijkstra's shortest path algorithm (Righini and Salani 2004)

The running time is dependent on the 'length of the path', i.e a function of the area of the biggest cycle in the figure

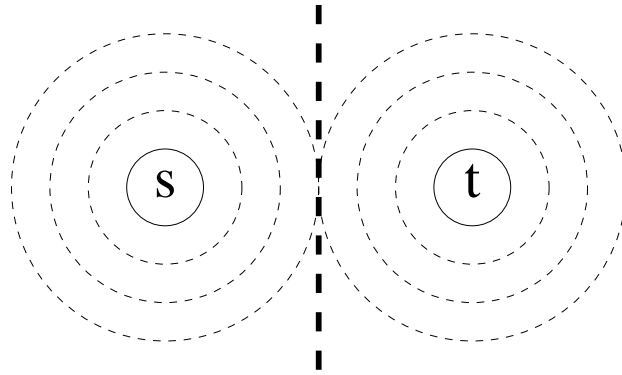


Speed up algorithm by making it bi-directional

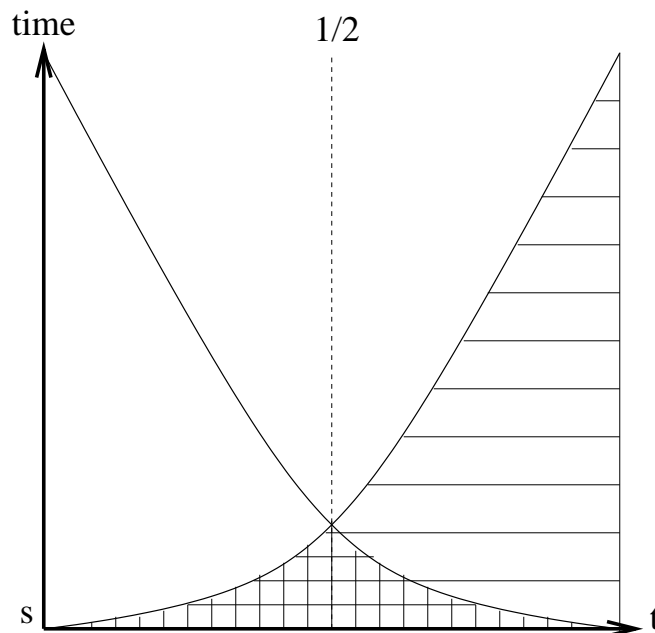
- Start two paths. One going from s towards t and another going in the opposite direction from t
- Splice the paths at 'the middle'
- Not used with k -cyc-SPPRC before

Bi-directionality (continued)

The running time is now a function of the areas of the two biggest cycles



The function can be polynomial or exponential depending on the grade of relaxation



Bi-directionality (continued)

Concepts

Motivation:

- Number of labels dependent on length of path
- Half the length of sub-paths

The bi-directional algorithm consist of the following three parts:

- Finding the shortest forward path from s to t the same time as finding the shortest backward ‘reverse path’ from t to s
- Combine a forward label L_f and backward label L_b with $v(L_f) = v(L_b)$ to obtain a path $P(L_f, L_b)$
- Stop at the ‘middle’, e.g., stop when the consumption of resource r_m in a label reaches $x_{stop} = \lceil b_{r_m}(t)/2 \rceil$

If lack of monotone resource r_m can be the number of visited nodes

Bi-directionality (continued)

Start one forward label L_f (direction $s \rightsquigarrow t$) in node s and one backward label L_b (direction $t \rightsquigarrow s$) in node t

First forward label is initialized as in monodirectional algorithm to lower bound of node s , i.e., $r(L_f) = a_r(s), \forall r \in R$

First backward label is initialized to upper bound of resources in node t , i.e., $r(L_b) = b_r(t), \forall r \in R$

Update backward-nodes reverse of forward-nodes

Let L'_f and L'_b represent some later extension of L_f and L_b

Stop extending labels when ‘the middle’ reached e.g. when $r(L'_b) \leq b_r(t)/2$ or $r(L'_f) > b_r(t)/2$ for some monotone resource $r_m \in R$

Bi-directionality (continued)

Splice forward label L'_f and backward label L'_b somewhere unique when:

$$\bar{v}(L'_f) = \bar{v}(L'_b) \quad (24)$$

$$L'_b \in \mathcal{E}(L'_f) \quad (25)$$

Somewhere unique could be when L'_b at ‘the middle’ or at node s

$L'_b \in \mathcal{E}(L'_f)$ is equivalent to

$$r(L'_f) \leq r(L'_b) \quad \forall r \in R \quad (26)$$

$$\bar{\mathcal{E}}_i(L'_f) \neq \bar{\mathcal{E}}_j(L'_b) \quad \forall i + j < k \quad (27)$$

Bi-directionality (continued)

Pseudocode

```
BI-DIRECTIONAL-LABEL-SETTING( $G, s, t$ )
1   $L_f \leftarrow$  FIRST-LABEL-FORWARD( $s$ )
2   $L_b \leftarrow$  FIRST-LABEL-BACKWARD( $t$ )
3   $PQ_f$ .ENQUEUE( $L_f$ )
4   $PQ_b$ .ENQUEUE( $L_b$ )
5  while  $PQ_f \neq \emptyset$  or  $PQ_b \neq \emptyset$ 
6      do if  $PQ_f$ .MIN().VALUE()  $<$   $x_{stop} - PQ_b$ .MAX().VALUE()
7          then REMOVE-DOMINATED( $PQ_f$ )
8               $L \leftarrow PQ_f$ .DEQUEUE()
9          else REMOVE-DOMINATED( $PQ_b$ )
10              $L \leftarrow PQ_b$ .DEQUEUE()
11         for each node  $v \in$  EXTENDABLES( $L$ )
12             do  $L_v \leftarrow$  EXTEND-LABEL( $L, v$ )
13                 for each label  $L \in$  SPLICEABLE( $L_v$ )
14                     do  $path \leftarrow$  SPLICE( $L_v, L$ )
15                         STORE-SOLUTION( $path, sol$ )
16 return  $sol$ 
```

Bi-directionality (continued)

Two separate queues. One for forward labels and one for backward labels

Forward extends labels to the ‘middle’ or directly across

Backward extends labels to the ‘middle’ but no further

Extendables(L) secures that labels that have reached the ‘middle’ are not extended further

Procedure $\text{Extend-Label}_{forward}(L, v)$ creates a new label L_v from L where

- $v(L_v) = v,$
- $p(L_v) = L,$
- $r(L_v) = r(L) + c_r(\bar{v}(L), v) \quad \forall r \in R$

Procedure $\text{Extend-Label}_{backward}(L, v)$ creates a new label L_v from L where

- $v(L_v) = v,$
- $p(L_v) = L,$
- $r(L_v) = r(L) - c_r(v, \bar{v}(L)) \quad \forall r \in R$

Bi-directionality (continued)

The procedure $\text{Store-Solution}_{forward}(L, sol)$ checks L against all backward labels L_i with $v(L_i) = v(L)$. If $L_i \in \mathcal{E}(L)$ the two labels combined is a feasible path

The procedure $\text{Store-Solution}_{backward}(L, sol)$ checks L against all forward labels L_i with $v(L_i) = v(L)$. If $L \in \mathcal{E}(L_i)$ the two labels combined is a feasible path

$L_i \in \mathcal{E}(L_j)$ when $r(L_j) \leq r(L_i) : \forall r \in R$ and the sets of recently visited nodes $\pi(L_i)$ and $\pi(L_j)$ fit each other

Bi-directionality (continued)

Unique splicing

A path $p = v_1 \rightarrow \dots \rightarrow v_n$ can potentially be spliced at several nodes $v_i : 1 \leq i \leq n$

It is desirable to find each path only once

Any way of choosing the node uniquely will do

If more than x_{stop} of resource r_m is consumed on path p ($r_m(p) \geq x_{stop}$), one edge $e(i, j) \in p$ either crosses x_{stop} or ends at x_{stop} . Choosing node j will be unique for p

If $r_m(p) < x_{stop}$ choosing the first (or last) node of p will be unique

Bi-directionality (continued)

Proof of correctness

Wlog let path $P = v_1 \rightarrow \dots \rightarrow v_n$ be optimal and unique

For all $v_i \in P$:

L_f^i) label at v_i on P for forward algorithm

L_b^i) label at v_i on P for backward algorithm

Proof by contradiction. Assume that P is not found. This can only happen in three cases:

- 1) For some node $v \in V(P)$ neither L_f^v nor L_b^v is created.
- 2) For some node $v \in V(P)$ neither L_f^v nor L_b^v exist after domination.
- 3) There is no node $v \in V(P)$ where both L_f^v and L_b^v exist after domination.

It will now be show that none of the three cases can happen

Bi-directionality (continued)

Case 1: Both forward and backward algorithm find P , so stopping criteria must have stopped both of them before node v was reached:

$$r_{mono}(L_f^v) > x_{stop} > r_{mono}(L_b^v) \Rightarrow L_f^v \notin E(L_b^v)$$

which contradict that P is feasible

Case 2: At least one of L_f^v and L_b^v must have been removed during domination, which is in contradiction with Definition 4 or that P is unique and optimal

Case 3: Divides into two case: one where $r_{mono}(P) \leq x_{stop}$ and another where $r_{mono}(P) > x_{stop}$.

- 1) If $r_{mono}(P) \leq x_{stop}$, then the splicing must be done at v_n . L_b^n clearly exist, so L_f^n must be absent. This can only happen when $r_{mono}(L_f^{n-1}) > x_{stop}$ which contradict that $r_{mono}(P) \leq x_{stop}$
- 2) If $r_{mono}(P) > x_{stop}$, then there must be a node $v_i \in V(P)$ where L_b^i cannot be extended more due to $r_{mono}(L_b^i) - r_{mono}(e(v_{i-1}, v_i)) = r_{mono}(L_b^{i-1}) < x_{stop}$. Since L_f^i does not exist, $r_{mono}(L_f^{i-1}) > x_{stop}$. This means that $r_{mono}(L_f^{i-1}) > x_{stop} > r_{mono}(L_b^{i-1}) \Rightarrow L_f^{i-1} \notin E(L_b^{i-1})$ which contradicts that P is feasible

Test Data

Different relaxations

$$S = \{v_i \in V \mid i \bmod 2 \neq 0\}$$

RC206.100-it125				
SPPRC type	t(mono)	t(bi)	value	revisits
SPPRC	14	14	-4519.5	45
2-cyc-SPPRC	28	17	-1624.5	30
3-cyc-SPPRC	64	30	-797.5	16
4-cyc-SPPRC	410	152	-507.5	11
5-cyc-SPPRC	9586	3434	-269.0	8
3-cyc-SPPRC & S	154	90	-228.5	9

Test Data (continued)

Bi-directional

Instance	Type	L Mono	L BI	T Mono	T BI
r206-i1	ESPPRC	77546	41435	17.39	12.68
r206-i2	ESPPRC	243204	96942	304.67	65.87
r206-i3	ESPPRC	159785	69437	109.27	38.31
r206-i4	ESPPRC	149572	66405	92.91	36.65
r206-i5	ESPPRC	152166	66873	100.10	37.57
r206-i6	ESPPRC	150864	66904	96.63	37.56
r206-i7	ESPPRC	169239	71288	130.43	45.17
c203-i1	ESPPRC	159655	57619	107.22	10.27
c203-i2	ESPPRC	220308	63737	222.76	12.50
c203-i3	ESPPRC	119710	40551	31.73	6.82
c203-i4	ESPPRC	81265	29904	16.23	3.37
c203-i5	ESPPRC	150753	45410	77.20	6.40
c203-i6	ESPPRC	69351	27717	6.71	2.51
c203-i7	ESPPRC	71317	27863	7.51	2.38
A-n44-k6-i39	$k = 2$	4259	3008	0.80	0.56
A-n44-k6-i63	$k = 3$	8558	5868	3.01	1.53
A-n45-k7-i65	$k = 2$	13287	7427	4.08	1.91
A-n45-k7-i55	$k = 3$	12290	8441	4.61	2.48
A-n53-k7-i87	$k = 2$	11141	7447	3.93	2.02
A-n53-k7-i107	$k = 3$	17020	12897	12.69	5.61
B-n45-k6-i72	$k = 2$	43341	19141	79.85	10.29
B-n45-k6-i89	$k = 3$	54738	33469	126.98	32.70
P-n55-k15-i29	$k = 2$	7843	7713	2.01	1.64
P-n55-k15-i32	$k = 3$	9839	9576	3.31	2.50
P-n60-k10-i45	$k = 2$	5995	3879	1.09	0.67
P-n60-k10-i53	$k = 3$	18303	8985	5.43	1.81

Literature overview

Beasley, J.E., N. Christodes. 1989. An algorithm for the resource constrained shortest path problem. *Networks* 19 379–394.

Chabrier, A. 2005. Vehicle routing problem with elementary shortest path based column generation. In press: *Computers and Operations Research*.

Danna, E., C. Le Pape. 2005. Accelerating branch-and-price with local search: A case study on the vehicle routing problem with time windows, chap. 3 in *Column Generation*, G. Desaulniers, J. Desrosiers, and M. M. Solomon (editors). Kluwer Academic Publishers, 90–130.

Dumitrescu, I. 2002. Constrained path and cycle problems. Ph.D. thesis, Department of Mathematics and Statistics, University of Melbourne, Australia.

Feillet, D., P. Dejax, M. Gendreau, C. Gueguen. 2004. An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. *Networks* 44(3) 216–229.

Literature overview (continued)

Irnich, S., D. Villeneuve. 2006. The shortest path problem with resource constraints and k -cycle elimination for $k \geq 3$. Forthcoming in: INFORMS Journal of Computing.

Jepsen, M., B. Petersen, S. Spoorendonk, and D. Pisinger. 2006. A non-robust branch-and-cut-and-price algorithm for the vehicle routing problem with time windows. DIKU Department of Computer Science, University of Copenhagen, Denmark.

Righini, G., M. Salani. 2004. Symmetry helps: bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints. Tech. Rep. 66, Note del Polo - Ricerca, Dipartimento di Tecnologie dell'Informazione, Università degli studi di Milano. Submitted to Discrete Applied Mathematics.

Assignments

Ex1) Find another criteria for identifying a unique node to splice at

Ex2) Referee the article