An enhanced exact solution approach for the Constrained Shortest Path Tour Problem

D. Ferone[1], P. Festa[1], F. Guerriero[2]

September 7th, 2016

[1]*Department of Mathematics and Applications, University of Napoli, Federico II*
[2]*Department of Mechanical, Energetic and Management Engineering, University of Calabria*

Shortest Path Tour Problem

Let $G = (V, A, C)$ be a directed graph, where

- $V = \{1, \ldots, n\}$ is a set of nodes;
- $A = \{(i, j) \in V \times V \mid i, j \in V \wedge i \neq j\}$ is a set of m arcs;
- $C : A \to \mathbb{R}^+ \cup \{0\}$ is a function that assigns a nonnegative length c_{ij} to each arc $(i, j) \in A$;

Definition

The SPTP consists in finding a shortest path from a source node s to a destination node d, by ensuring that at least one node of each node subset $T_1, \ldots, T_N$, where $T_h \cap T_l = \emptyset$, $\forall\, h, l = 1, \ldots, N$, $h \neq l$, is crossed according to the sequence wherewith the subsets are ordered.

Constrained Shortest Path Tour Problem

An integer capacity $u_{ij} \geq 1$ is associated with each arc $(i, j) \in A$. It denotes the maximum number of times that arc (i, j) can be traversed in any CSPTP solution.

Theorem

*If $u_{ij} = 1$ for all $(i, j) \in A$, the resulting CSPTP is **NP**-hard.*

Hamiltonian Path problem (HAM-PATH) $\leq_m^p$ CSPTP.

Reduction

HAM-PATH	$\langle G = (V, A, C), s, d \rangle$
CSPTP	$\langle G' = (V', A', C'), s^-, d^+, \{T_h\}_{h=1,\ldots,n+1} \rangle$

Reduction

HAM-PATH	$\langle G = (V, A, C), s, d \rangle$
CSPTP	$\langle G' = (V', A', C'), s^-, d^+, \{T_h\}_{h=1,\dots,n+1} \rangle$

- for each node $i \in V$,
  - insert in V' nodes i^- and i^+;
  - insert in A' arc (i^-, i^+) with cost 0;

Reduction

HAM-PATH	$\langle G = (V, A, C), s, d\rangle$
CSPTP	$\langle G' = (V', A', C'), s^-, d^+, \{T_h\}_{h=1,\dots,n+1}\rangle$

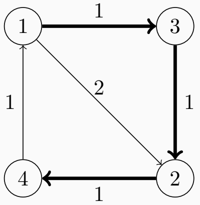
- for each node $i \in V$,
  - insert in V' nodes i^- and i^+;
  - insert in A' arc (i^-, i^+) with cost 0;
- for each arc $(i, j) \in A$ and for each $k = 2, \dots, |V|$,
  - insert in V' node ij^k;
  - insert in T_k node ij^k;
  - insert in A' arc (i^+, ij^k) with cost c_{ij} and arc (ij^k, j^-) with cost 0;

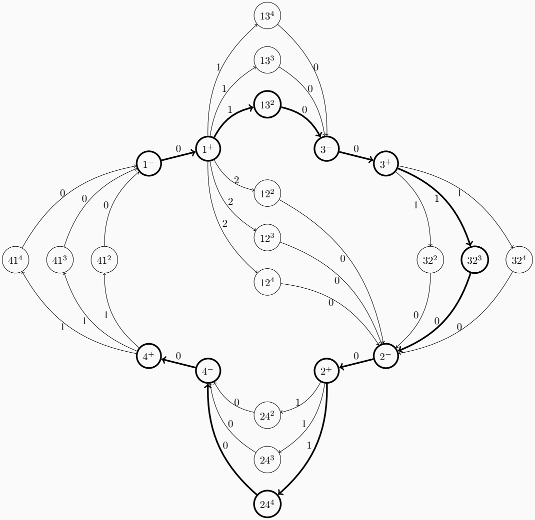
Reduction

HAM-PATH	$\langle G = (V, A, C), s, d \rangle$
CSPTP	$\langle G' = (V', A', C'), s^-, d^+, \{T_h\}_{h=1,\ldots,n+1} \rangle$

- for each node $i \in V$,
  - insert in V' nodes i^- and i^+;
  - insert in A' arc (i^-, i^+) with cost 0;
- for each arc $(i, j) \in A$ and for each $k = 2, \ldots, |V|$,
  - insert in V' node ij^k;
  - insert in T_k node ij^k;
  - insert in A' arc (i^+, ij^k) with cost c_{ij} and arc (ij^k, j^-) with cost 0;
- set $T_1 = \{s^-\}$ and $T_{|V|+1} = \{d^+\}$.

Reduction


1
3
2
4
1
1
1
1
2


13^4
13^3
13^2
1^-
1^+
3^-
3^+
12^2
12^3
12^4
41^4
41^3
41^2
32^2
32^3
32^4
4^+
4^-
2^+
2^-
24^2
24^3
24^4

Path existence

Lemma (3)

There exists a feasible path $P = i_1, i_2, \ldots, i_k$, $k \leq n$, *in*

$$\langle G = (V, A, C), s, d \rangle \,,$$

if and only if in

$$\left\langle G' = (V', A', C'), s^-, d^+, \{T_h\}_{h=1,\ldots,n+1} \right\rangle$$

there exists a path tour P' *from* i_1^- *to* i_k^+*, such that*

$$P' = \left\{ \bigoplus_{l=1}^{k-1} \left(i_l^-, i_l^+, {i_l i_{l+1}}^{l+1} \right), i_k^-, i_k^+ \right\}.$$

Proof.

$\Rightarrow$ Suppose that there exists in G a feasible path $P = \{i_1, i_2, \ldots, i_k\}$, $k \leq n$. Then, by construction there exists in A' an arc (i_l^-, i_l^+), for each $l = 1, \ldots, k$.

Path existence

Proof.
$\Rightarrow$ Suppose that there exists in G a feasible path $P = \{i_1, i_2, \ldots, i_k\}$, $k \leq n$. Then, by construction there exists in A' an arc (i_l^-, i_l^+), for each $l = 1, \ldots, k$.

Moreover, for each arc (i_l, i_{l+1}) in P, there exist arcs $(i_l^+, i_l i_{l+1}{}^q)$ and $(i_l i_{l+1}{}^q, i_{l+1}^-)$ for each $q = 2, \ldots, n$.

Path existence

Proof.
$\Rightarrow$ Suppose that there exists in G a feasible path $P = \{i_1, i_2, \ldots, i_k\}$, $k \leq n$. Then, by construction there exists in A' an arc (i_l^-, i_l^+), for each $l = 1, \ldots, k$.

Moreover, for each arc (i_l, i_{l+1}) in P, there exist arcs $(i_l^+, i_l i_{l+1}{}^q)$ and $(i_l i_{l+1}{}^q, i_{l+1}^-)$ for each $q = 2, \ldots, n$.

Therefore, there must exist also arcs $(i_l^+, i_l i_{l+1}{}^{l+1})$ and $(i_l i_{l+1}{}^{l+1}, i_{l+1}^-)$.

Path existence

Proof.
$\Leftarrow$ Conversely, suppose that there exists in G' the path P', whereas path P is not present in G. This last situation occurs if either at least one node $i_l \notin V$ or at least one arc $(i_l, i_{l+1}) \notin A$.

If a node $i_l \notin V$, then nodes i_l^- and i_l^+ would not be in V', which is not true. Similarly, if an arc $(i_l, i_{l+1}) \notin A$, then arcs $(i_l^+, i_l {i_{l+1}}^{l+1})$ and $(i_l {i_{l+1}}^{l+1}, i_{l+1}^-)$ would not be in A' and this contradicts the hypothesis of existence of path P'.

Proof

Theorem

The described procedure is a polynomially computable function $f()$ that transforms any instance $\mathcal{I}_{\text{HAM-PATH}}$ of HAM-PATH in an instance $\mathcal{I}_{\text{CSPTP}}$ of the CSPTP.

Existing approach

The old B&B

The CSPTP can be reduced to the Path Avoiding Forbidden Pairs Problem (PAFPP), where $F \subseteq \{ (v, w) \in V \times V \mid v \neq w \}$:

$$\min \sum_{(i,j) \in A} c_{ij} x_{ij}$$

s.t.

$$\sum_{j \in FS(i)} x_{ij} - \sum_{j \in BS(i)} x_{ji} = \begin{cases} 1, & i = s; \\ -1, & i = d; \\ 0, & \text{otherwise}; \end{cases}$$

$$\sum_{j \in BS(a)} x_{ja} + \sum_{j \in BS(b)} x_{jb} \leq 1 \qquad \forall\, (a, b) \in F$$

$$x_{ij} \in \{ 0, 1 \} \qquad \forall\, (i, j) \in A.$$

The old B&B

The CSPTP can be reduced to the Path Avoiding Forbidden Pairs Problem (PAFPP), where $F \subseteq \{\, (v, w) \in V \times V \mid v \neq w \,\}$:

$$\min \sum_{(i,j)\in A} c_{ij} x_{ij}$$

s.t.

$$\sum_{j\in FS(i)} x_{ij} - \sum_{j\in BS(i)} x_{ji} = \begin{cases} 1, & i = s; \\ -1, & i = d; \\ 0, & \text{otherwise}; \end{cases}$$

$$\cancel{\sum_{j\in BS(a)} x_{ja} + \sum_{j\in BS(b)} x_{jb} \leq 1} \qquad \cancel{\forall\,(a,b) \in F}$$

$$x_{ij} \in \{\, 0, 1 \,\} \qquad \forall\,(i,j) \in A.$$

The old B&B

At a generic iteration t:

- solve the relaxed Shortest Path Problem (e.g. Dijkstra);

The old B&B

At a generic iteration t:

- solve the relaxed Shortest Path Problem (e.g. Dijkstra);
- if does not contain any forbidden pair, is feasible for CSPTP;

The old B&B

At a generic iteration t:

- solve the relaxed Shortest Path Problem (e.g. Dijkstra);
- if does not contain any forbidden pair, is feasible for CSPTP;
- else, let (a_k, b_k) be a violated forbidden pair;

The old B&B

At a generic iteration t:

- solve the relaxed Shortest Path Problem (e.g. Dijkstra);
- if does not contain any forbidden pair, is feasible for CSPTP;
- else, let (a_k, b_k) be a violated forbidden pair;
- generate two subproblems: in the first one remove the node a_k, in the second one remove the node b_k.

The old B&B

At a generic iteration t:

- solve the relaxed Shortest Path Problem (e.g. Dijkstra);
- if does not contain any forbidden pair, is feasible for CSPTP;
- else, let (a_k, b_k) be a violated forbidden pair;
- generate two subproblems: in the first one remove the node a_k, in the second one remove the node b_k.

Very simple, but the preprocessing time is too long!

New approach

Main ideas

- Solve the problem on the original graph;

Main ideas

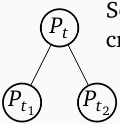
- Solve the problem on the original graph;
- a path tour is a concatenation of simple paths $T_i \rightsquigarrow T_{i+1}$ for $i = 1, \ldots, N-1$;

Main ideas

- Solve the problem on the original graph;
- a path tour is a concatenation of simple paths $T_i \rightsquigarrow T_{i+1}$ for $i = 1, \ldots, N-1$;
- an arc repetition can occur only in two different subpaths $T_i \rightsquigarrow T_{i+1}$ and $T_j \rightsquigarrow T_{j+1}$, with $i \neq j$.

Main ideas

- Solve the problem on the original graph;
- a path tour is a concatenation of simple paths $T_i \rightsquigarrow T_{i+1}$ for $i = 1, \ldots, N-1$;
- an arc repetition can occur only in two different subpaths $T_i \rightsquigarrow T_{i+1}$ and $T_j \rightsquigarrow T_{j+1}$, with $i \neq j$.



Solution infeasible because (v, w) crossed both in $T_i \rightsquigarrow T_{i+1}$ and $T_j \rightsquigarrow T_{j+1}$

Impose solution does not contain (v, w) in $T_i \rightsquigarrow T_{i+1}$

Impose solution does not contain (v, w) in $T_j \rightsquigarrow T_{j+1}$

The enhanced B&B^{new}

Function BB($G = \langle V, A, C \rangle, s, d, \{T_i\}_{i=1,\ldots,N}$)

ShortestPaths $\leftarrow$ FloydWarshall(G)

$x \leftarrow$ DP$(V, A, s, \{T_i\}_{i=1,\ldots,N},)$

if x *is feasible* **then**

  return $(x, z(x))$

The enhanced B&B^{new}

Function BB$(G = \langle V, A, C \rangle, s, d, \{T_i\}_{i=1,\dots,N})$

ShortestPaths $\leftarrow$ FloydWarshall(G)

$x \leftarrow$ DP$(V, A, s, \{T_i\}_{i=1,\dots,N},)$

if x *is feasible* **then**

  return $(x, z(x))$

for $i \leftarrow 1$ ***to*** $N-1$ **do**

  foreach $v \in T_i$ **do**

    foreach $w \in T_{i+1}$ **do**

      Paths$[i] \leftarrow$ Paths$[i] \cup \{$ShortestPaths$[v][w]\}$

$Q \leftarrow$ GenerateNodes$(x, Paths, [\emptyset]_{i=1}^{N-1})$

$x^* \leftarrow \mathbf{Nil}$; $z(x^*) \leftarrow +\infty$

The enhanced B&B^{new}

Function BB($G = \langle V, A, C \rangle, s, d, \{T_i\}_{i=1,\dots,N}$)

while Q *is not empty* **do**

  $Node \leftarrow \textsc{Pop}(Q)$

  $i \leftarrow Node.index$

  $A \leftarrow A \setminus Node.costraints[i]$

The enhanced B&B^{new}

Function BB($G = \langle V, A, C \rangle, s, d, \{T_i\}_{i=1,\ldots,N}$)

while Q *is not empty* **do**

  $Node \leftarrow \textsc{Pop}(Q)$

  $i \leftarrow Node.index$

  $A \leftarrow A \setminus Node.costraints[i]$

  foreach $v \in T_i$ **do**

    foreach $w \in T_{i+1}$ **do**

      $Node.paths[i] \leftarrow Node.paths[i] \cup \{\textsc{Dijkstra}(G, v, w)\}$

The enhanced B&B^{new}

Function BB($G = \langle V, A, C \rangle, s, d, \{T_i\}_{i=1,\dots,N}$)

while Q *is not empty* **do**

  $Node \leftarrow \textsc{Pop}(Q)$

  $i \leftarrow Node.index$

  $A \leftarrow A \setminus Node.costraints[i]$

  foreach $v \in T_i$ **do**

    foreach $w \in T_{i+1}$ **do**

      $Node.paths[i] \leftarrow Node.paths[i] \cup \{\textsc{Dijkstra}(G, v, w)\}$

  $x \leftarrow \mathrm{DP}(Node.paths)$

  $A \leftarrow A \cup Node.costraints[i]$

The enhanced B&B^{new}

Function BB($G = \langle V, A, C \rangle, s, d, \{T_i\}_{i=1,\dots,N}$)

while Q *is not empty* **do**
  $Node \leftarrow$ Pop(Q)
  $i \leftarrow Node.index$
  $A \leftarrow A \setminus Node.costraints[i]$
  foreach $v \in T_i$ **do**
    foreach $w \in T_{i+1}$ **do**
      $Node.paths[i] \leftarrow Node.paths[i] \cup \{$Dijkstra$(G, v, w)\}$
  $x \leftarrow$ DP($Node.paths$)
  $A \leftarrow A \cup Node.costraints[i]$
  if x *is feasible* **then**
    if $z(x) < z(x^*)$ **then**
      $x^* \leftarrow x, z(x^*) \leftarrow z(x)$
  else if $z(x) < z(x^*)$ **then**
    $Q \leftarrow Q \cup$ GenerateNodes($x, Node.paths, Node.constraints$)
return $(x^*, z(x^*))$

The enhanced B&B^{new}

Function GenerateNodes($x, paths, contraints$)

$((v,w), i, j) \leftarrow$ FIND(x)

$Node_1 \leftarrow$ GENERATENODE($paths, contraints, i, v, w$)

$Node_2 \leftarrow$ GENERATENODE($paths, contraints, j, v, w$)

return $\{Node_1, Node_2\}$

The enhanced B&B^{new}

Function GenerateNodes($x, paths, contraints$)

$((v, w), i, j) \leftarrow \textsc{Find}(x)$

$Node_1 \leftarrow \textsc{GenerateNode}(paths, contraints, i, v, w)$

$Node_2 \leftarrow \textsc{GenerateNode}(paths, contraints, j, v, w)$

return $\{Node_1, Node_2\}$

Function GenerateNode($paths, constraints, i, v, w$)

$Node.paths \leftarrow paths$

$Node.constraints \leftarrow constraints$

$Node.constraints[i] \leftarrow Node.constraints[i] \cup \{(v, w)\}$

$Node.paths[i] \leftarrow \emptyset$

$Node.index \leftarrow i$

return $Node$

Experimental results

Test environment

- Implemented in C++, and compiled with g++ (Ubuntu 5.2.1-22ubuntu2) 5.2.1 Flag: -std=c++14;
- running times reported are UNIX real wall-clock times in seconds.
- experiments were run on S.Co.P.E. (Unina), a cluster of nodes, each of them with two processors Intel Xeon E5-4610v2@2.30 Ghz.

Test problems

- Complete graphs with $n \in \{ 100, \dots, 1000 \}$ with a step of 50, the number N of sets $\{ T_i \}_{i=1}^{N}$ is $25\%n$ and $35\%n$ nodes belong to any T_i;

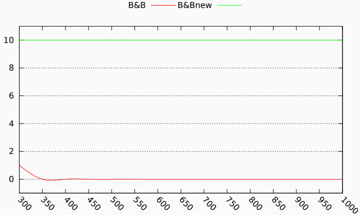
Test problems

- Complete graphs with $n \in \{ 100, \ldots, 1000 \}$ with a step of 50, the number N of sets $\{ T_i \}_{i=1}^{N}$ is $25\%n$ and $35\%n$ nodes belong to any T_i;
- Grid graphs in $\{5 \times 20, 7 \times 15, 9 \times 9, 10 \times 10, 10 \times 40,$ $14 \times 30\}$. Nodes belong to any T_i is $35\%n$, and $N \in \{ 15, 16, 17, 18, 19 \}$ (in percentage of n);

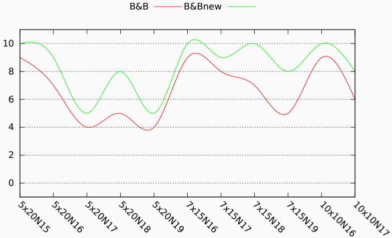
Test problems

- Complete graphs with $n \in \{ 100, \ldots, 1000 \}$ with a step of 50, the number N of sets $\{ T_i \}_{i=1}^{N}$ is $25\%n$ and $35\%n$ nodes belong to any T_i;
- Grid graphs in $\{5 \times 20, 7 \times 15, 9 \times 9, 10 \times 10, 10 \times 40,$ $14 \times 30\}$. Nodes belong to any T_i is $35\%n$, and $N \in \{ 15, 16, 17, 18, 19 \}$ (in percentage of n);
- Random graphs with $n \in \{250, 500, 750, 1000\}$ and $m \in \{10, 20, 30, 40, 50, 60, 70, 80, 90, 100\}\%n(n-1)$, $N = 25\%n$ and $35\%n$ nodes belong to any T_i.

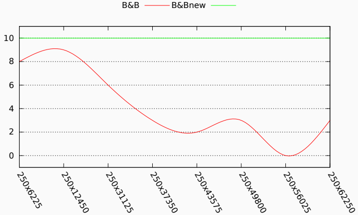
Complete



Grid



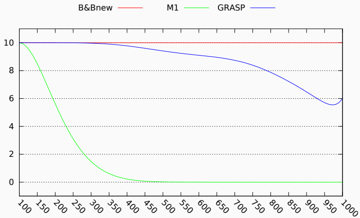
Random



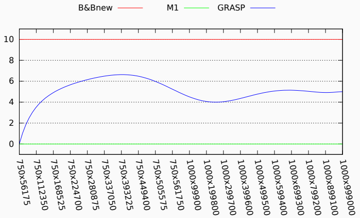
Performance profiles of BB and B&B^{new} algorithms for optimal solutions.

Empirical evalution

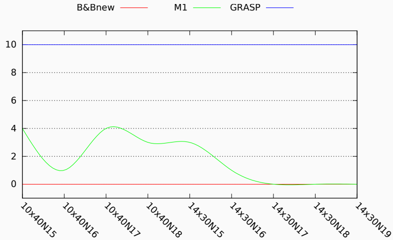
Exact complete



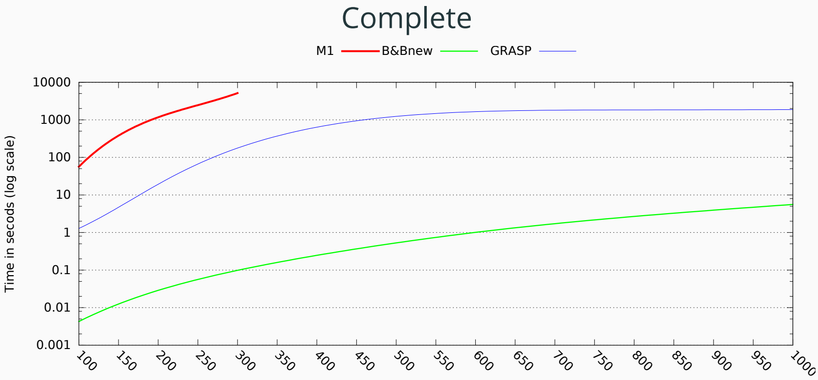
Exact random



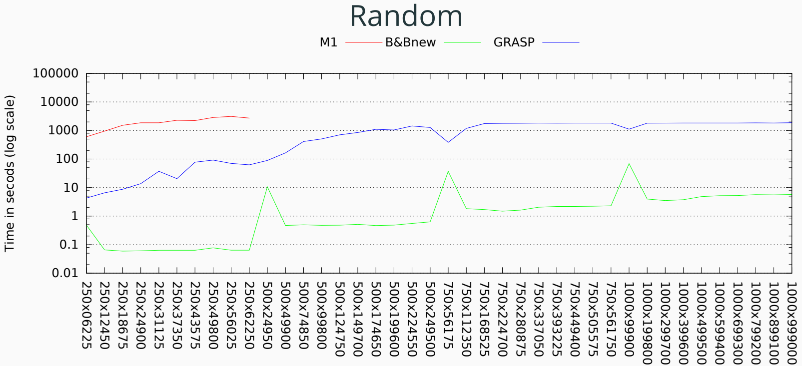
Feasible grid



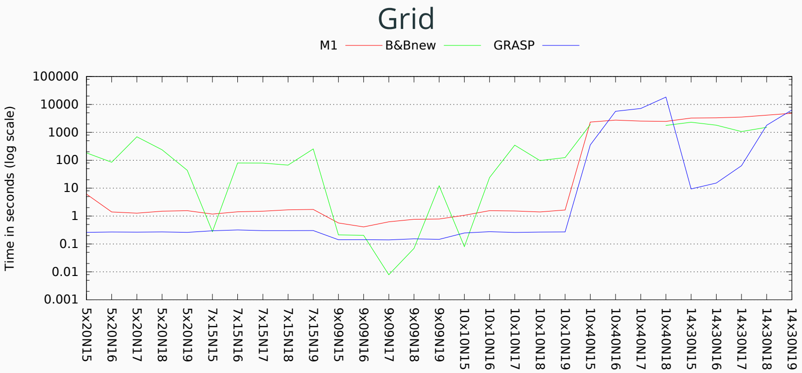
Time comparison



Time comparison



Time comparison



Conclusions and future work

- B&B^{new} improves the results of `BB`: it is faster and needs less memory;

Conclusions and future work

- B&B^{new} improves the results of BB: it is faster and needs less memory;
- it has very good performances expecially on dense graphs;

Conclusions and future work

- B&B^{new} improves the results of BB: it is faster and needs less memory;
- it has very good performances expecially on dense graphs;
- as future work, we are investigating further variants of the problem resulting from the introduction of further constraints defined on the arcs and/or on the nodes of the graph.

01000101 01001110 01000100

(E N D)

Thank you.

Proof

Theorem

The described procedure is a polynomially computable function $f()$ that transforms any instance $\mathcal{I}_{\textit{HAM-PATH}}$ of HAM-PATH in an instance $\mathcal{I}_{\textit{CSPTP}}$ of the CSPTP.

Proof

Theorem

The described procedure is a polynomially computable function $f()$ that transforms any instance $\mathcal{I}_{\textit{HAM-PATH}}$ of HAM-PATH in an instance $\mathcal{I}_{\textit{CSPTP}}$ of the CSPTP.

Proof.

$\Rightarrow$ By hypothesis, there exists in G a Hamiltonian path $P = \{i_1, i_2, \cdots, i_n\}$, where $i_1 = s$ and $i_n = d$. We have already shown in Lemma 3 that there exists in G' a path

$$P' = \left\{ \bigoplus_{l=1}^{n-1} \left(i_l^-, i_l^+, i_l i_{l+1}{}^{l+1} \right) i_n^-, i_n^+ \right\},$$

where $i_1^- = s^-$ and $i_n^+ = d^+$.

Proof

Theorem

The described procedure is a polynomially computable function $f()$ that transforms any instance $\mathcal{I}_{\textit{HAM-PATH}}$ of HAM-PATH in an instance $\mathcal{I}_{\textit{CSPTP}}$ of the CSPTP.

Proof.

$\Rightarrow$ P' is a feasible constrained path tour from s^- to d^+. In fact, let us suppose that P' is not feasible. This can happen if at least one of the following cases occurs: 1) P' crosses some arcs more than once; 2) P' does not involve any node in some node subsets T_i, $i = 1, \dots, n+1$; 3) P' involves at least a node for each T_i, , $i = 1, \dots, n+1$, but not successively and sequentially.

Proof

Theorem

The described procedure is a polynomially computable function $f()$ that transforms any instance $\mathcal{I}_{\text{HAM-PATH}}$ of HAM-PATH in an instance $\mathcal{I}_{\text{CSPTP}}$ of the CSPTP.

Proof.

$\Rightarrow$ Suppose that P' crosses some arcs twice. Since only nodes of type i^- are such that $|FS(i^-)| > 1$, if some arc is involved at least twice, it must be some arc of type (i^-, i^+). Nevertheless, if this is the case, then necessarily node i must be involved by P at least twice and this contradicts the hypothesis of P as Hamiltonian path.

Proof

Theorem

The described procedure is a polynomially computable function $f()$ that transforms any instance $\mathcal{I}_{\textit{HAM-PATH}}$ of HAM-PATH in an instance $\mathcal{I}_{\textit{CSPTP}}$ of the CSPTP.

Proof.

$\Rightarrow$ Finally, cases 2) and 3) can not ever occur by construction. In fact, path P' starts at $s^- \in T_1$ and ends in $d^+ \in T_{n+1}$. Then, it involves successively and sequentially all nodes ${i_l i_{l+1}}^{l+1}$, for each $l = 1, \ldots, n-1$, and each node ${i_l i_{l+1}}^{l+1}$ belongs to T_{l+1}.

Proof

$\Leftarrow$ By hypothesis, there exists in G' a feasible constrained path tour from s^- to d^+.

Remember that by construction, it holds that

- for each node $i^- \in V'$, $FS(i^-) = \{i^+\}$;
- for each node $i^+ \in V'$, $FS(i^+) = \{ij^k \mid k = 2, \ldots, n\}$;
- for each node $ij^k \in V'$, $FS(ij^k) = \{j^-\}$.

Therefore, path P' must be necessarily as follows

$$P' = \left\{ \bigoplus_{l=1}^{n-1} \left(i_l^-, i_l^+, {i_l i_{l+1}}^{l+1} \right) i_n^-, i_n^+ \right\},$$

where $i_1^- = s^-$ and $i_n^+ = d^+$.

Proof

$\Leftarrow$ In fact, if for some $k = 2, \ldots, n$, $k \neq l+1$, P' contains a subpath

$$i_l, i_{l+1}, i_l i_{l+1}{}^k,$$

then P' would not be feasible, because it would violate the constraint of successively and sequentially passing through at least one node of the node subsets T_i. Finally, if P' involves a smaller number of nodes, then for some subset T_i, no node in T_i would be crossed. Similarly, if P' involves a higher number of nodes, then P' would cross at least one arc more than once.

Proof

$\Leftarrow$ From Lemma 3, it follows that there exists in G a path $P = \{i_1, i_2, \ldots, i_n\}$, such that $i_1 = s$ and $i_n = d$.

P must be Hamiltonian. In fact, let us suppose that P is not Hamiltonian. Since P visits exactly n nodes, there must be i_j and i_k, $j, k \in \{1, \ldots, n \mid j \neq k\}$, such that $i_j = i_k$. But this implies that P' crosses arcs (i_j^-, i_j^+) and (i_k^-, i_k^+) such that $(i_j^-, i_j^+) \equiv (i_k^-, i_k^+)$ and this contradicts the hypothesis of feasibility of the constrained path tour P'.

The Hamiltonian path P in G and the constrained path tour P' in G' have the same length by construction and by the definition of cost functions C and C', respectively. $\square$