Probabilistic analysis of an approximation algorithm for the m-peripatetic salesman problem on random instances unbounded from above.

Edward Gimadi, Alexey Istomin, Ivan Rykov, Oxana Tsidulko

Sobolev Institute of Mathematics SB RAS
Novosibirsk State University

Definitions

An algorithm A has performance bounds $\varepsilon_A(n), \delta_A(n)$ if

$$\mathbf{Pr}\Big\{F_A > \big(1+\varepsilon_A(n)\big)OPT\Big\} \le \delta_A(n)$$

Definitions

An algorithm A has performance bounds $\varepsilon_A(n), \delta_A(n)$ if

$$\mathbf{Pr}\Big\{F_A > \big(1 + \varepsilon_A(n)\big)OPT\Big\} \le \delta_A(n)$$

- n is the problem size

Definitions

An algorithm A has performance bounds $\varepsilon_A(n), \delta_A(n)$ if

$$\mathbf{Pr}\Big\{F_A > \big(1 + \varepsilon_A(n)\big)OPT\Big\} \leq \delta_A(n)$$

- n is the problem size
- $\varepsilon_A(n)$ is the relative error of the algorithm

Definitions

An algorithm A has performance bounds $\varepsilon_A(n), \delta_A(n)$ if

$$\mathbf{Pr}\Big\{F_A > \big(1 + \varepsilon_A(n)\big)OPT\Big\} \leq \delta_A(n)$$

- n is the problem size
- $\varepsilon_A(n)$ is the relative error of the algorithm
- $\delta_A(n)$ is the failure probability of the algorithm, i.e. the proportion of cases when algorithm A doesn't hold the relative error $\varepsilon_A(n)$.

Definitions

An algorithm A has performance bounds $\varepsilon_A(n), \delta_A(n)$ if

$$\mathbf{Pr}\Big\{F_A > \big(1+\varepsilon_A(n)\big)OPT\Big\} \le \delta_A(n)$$

- n is the problem size
- $\varepsilon_A(n)$ is the relative error of the algorithm
- $\delta_A(n)$ is the failure probability of the algorithm, i.e. the proportion of cases when algorithm A doesn't hold the relative error $\varepsilon_A(n)$.

An algorithm A is called asymptotically optimal (exact)

on a class of instances, if there exist performance bounds s.t.

$$\varepsilon_A(n) \xrightarrow[n\to\infty]{} 0, \quad \delta_A(n) \xrightarrow[n\to\infty]{} 0.$$

Discrete optimization problems on graphs often consist in finding a subgraph of extreme total weight: e.g. a spanning tree, a perfect matching, a hamiltonian cycle, etc.

Discrete optimization problems on graphs often consist in finding a subgraph of extreme total weight: e.g. a spanning tree, a perfect matching, a hamiltonian cycle, etc.

Some of this problems are polynomially solvable, like

- the Assignment Problem (E.A. Dinic, M.A. Kronord)
- the Minimum Spanning Tree Problem (Prim).

Discrete optimization problems on graphs often consist in finding a subgraph of extreme total weight: e.g. a spanning tree, a perfect matching, a hamiltonian cycle, etc.

Some of this problems are polynomially solvable, like

- the Assignment Problem (E.A. Dinic, M.A. Kronord)
- the Minimum Spanning Tree Problem (Prim).

But most of this problems are NP-hard, like the well-known Travelling Salesman Problem.

m-Peripatetic Salesman Problem (m-PSP)[Krarup 1974]

The problem is to find

m edge-disjoint Hamiltonian cycles $H_1, \ldots, H_m$
in a given complete graph $G = (V, E)$
with given weight functions $w_i : E \to \mathbf{R}_+$, $i = 1, \ldots, m$,

such that

$$W_1(H_1) + \ldots + W_m(H_m) = \sum_{i=1}^{m} \sum_{e \in H_i} w_i(e) \to \min(\max).$$

m-Peripatetic Salesman Problem (m-PSP)[Krarup 1974]

The problem is to find

m edge-disjoint Hamiltonian cycles $H_1, \ldots, H_m$
in a given complete graph $G = (V, E)$
with given weight functions $w_i : E \to \mathbf{R}_+$, $i = 1, \ldots, m$,

such that

$$W_1(H_1) + \ldots + W_m(H_m) = \sum_{i=1}^{m} \sum_{e \in H_i} w_i(e) \to \min(\max).$$

The problem is NP-hard

Applications of the m-PSP include

Design of patrol tours

in order to avoid constantly repeating the same tour and thus enhance the security.

Applications of the m-PSP include

Design of patrol tours

in order to avoid constantly repeating the same tour and thus enhance the security.

Network design applications with high level of data transmission reliability:

in order to protect the network from link failure, several edge-disjoint cycles need to be determined.

Applications of the *m*-PSP include

Design of patrol tours

in order to avoid constantly repeating the same tour and thus enhance the security.

Network design applications with high level of data transmission reliability:

in order to protect the network from link failure, several edge-disjoint cycles need to be determined.

Scheduling the machine processing:

e.g. scheduling application where each job must be processed twice by the same machine but technological constraints prevent the repetition of identical job sequences.

Applications of the m-PSP include

Design of patrol tours

in order to avoid constantly repeating the same tour and thus enhance the security.

Network design applications with high level of data transmission reliability:

in order to protect the network from link failure, several edge-disjoint cycles need to be determined.

Scheduling the machine processing:

e.g. scheduling application where each job must be processed twice by the same machine but technological constraints prevent the repetition of identical job sequences.

Optimization of delivery routes.

The m-PSP is studied in the cases of

- deterministic and random instances,

The m-PSP is studied in the cases of

- deterministic and random instances,
- arbitrary, Euclidean and metric weight functions of edges,

The m-PSP is studied in the cases of

- deterministic and random instances,
- arbitrary, Euclidean and metric weight functions of edges,
- common and different weight functions of m Hamiltonian cycles

The m-PSP is studied in the cases of

- deterministic and random instances,
- arbitrary, Euclidean and metric weight functions of edges,
- common and different weight functions of m Hamiltonian cycles
- special classes of graphs where the weights of the edges belong to a given finite and infinite set of numbers.

Known results for the m-PSP

Some previous results for m-PSP

- NP-hardness [De Kort 1991].

Some previous results for m-PSP

- NP-hardness [De Kort 1991].
- Polyn. solvable cases of 2-PSP [De Brey&Volgenant 1997].

Some previous results for m-PSP

- NP-hardness [De Kort 1991].
- Polyn. solvable cases of 2-PSP [De Brey&Volgenant 1997].
- LB and UB for 2-PSP in B&B [De Kort,1991-93].

Known results for the m-PSP

Some previous results for m-PSP

- NP-hardness [De Kort 1991].
- Polyn. solvable cases of 2-PSP [De Brey&Volgenant 1997].
- LB and UB for 2-PSP in B&B [De Kort,1991-93].
- Polyhedral approach for m-PSP [Duch&Lapor&Sem 2005].

Known results for the *m*-PSP

Some previous results for *m*-PSP

- NP-hardness [De Kort 1991].
- Polyn. solvable cases of 2-PSP [De Brey&Volgenant 1997].
- LB and UB for 2-PSP in B&B [De Kort,1991-93].
- Polyhedral approach for *m*-PSP [Duch&Lapor&Sem 2005].

Some Novosibirsk group results for *m*-PSP

Known results for the *m*-PSP

Some previous results for *m*-PSP

- NP-hardness [De Kort 1991].
- Polyn. solvable cases of 2-PSP [De Brey&Volgenant 1997].
- LB and UB for 2-PSP in B&B [De Kort,1991-93].
- Polyhedral approach for *m*-PSP [Duch&Lapor&Sem 2005].

Some Novosibirsk group results for *m*-PSP

- Polynomial approximation algorithms with performance guarantees for 2-PSP (2004-2012, Ageev, Baburin, Gimadi, Glazkov, Glebov, Korkishko, Pyatkin, Zambalaeva).

Known results for the *m*-PSP

Some previous results for *m*-PSP

- NP-hardness [De Kort 1991].
- Polyn. solvable cases of 2-PSP [De Brey&Volgenant 1997].
- LB and UB for 2-PSP in B&B [De Kort,1991-93].
- Polyhedral approach for *m*-PSP [Duch&Lapor&Sem 2005].

Some Novosibirsk group results for *m*-PSP

- Polynomial approximation algorithms with performance guarantees for 2-PSP (2004-2012, Ageev, Baburin, Gimadi, Glazkov, Glebov, Korkishko, Pyatkin, Zambalaeva).
- Polynomial asymptotically exact algorithms for m-PSP$_{max}$ in Euclidean space (Baburin&Gimadi- 2008-2010),

Known results for the *m*-PSP

Some previous results for *m*-PSP

- NP-hardness [De Kort 1991].
- Polyn. solvable cases of 2-PSP [De Brey&Volgenant 1997].
- LB and UB for 2-PSP in B&B [De Kort,1991-93].
- Polyhedral approach for *m*-PSP [Duch&Lapor&Sem 2005].

Some Novosibirsk group results for *m*-PSP

- Polynomial approximation algorithms with performance guarantees for 2-PSP (2004-2012, Ageev, Baburin, Gimadi, Glazkov, Glebov, Korkishko, Pyatkin, Zambalaeva).
- Polynomial asymptotically exact algorithms for m-PSP$_{max}$ in Euclidean space (Baburin&Gimadi- 2008-2010),
- Polyhedral space with a bounded number of facets (Shenmaier - 2010)

The results given in this presentation:

The results given in this presentation:

- We offer an approximation polynomial algorithm for the minimum-weight m-PSP.

The results given in this presentation:

- We offer an approximation polynomial algorithm for the minimum-weight m-PSP.
- We have obtained the performance guarantees of this algorithm for certain classes of random inputs of the problem.

The results given in this presentation:

- We offer an approximation polynomial algorithm for the minimum-weight m-PSP.
- We have obtained the performance guarantees of this algorithm for certain classes of random inputs of the problem.
- We have justified the conditions for the algorithm to be asymptotically exact on the considered classes of inputs.

Algorithm $\widetilde{A}$ for minimum-weight m-PSP

Input:

A complete n-vertex graph $G = (V, E)$ with weight functions $w_i : E \to \mathbf{R}_+, \; i = 1, \ldots, m$, where $m < n/4$

Algorithm $\widetilde{A}$ for minimum-weight m-PSP

Input:

A complete n-vertex graph $G = (V, E)$ with weight functions $w_i : E \to \mathbf{R}_+, \;\; i = 1, \ldots, m$, where $m < n/4$

Output:

m edge disjoint Hamiltonian cycles $H_1, \ldots, H_m$

Algorithm $\widetilde{A}$ for minimum-weight m-PSP

Input:

A complete n-vertex graph $G = (V, E)$ with weight functions $w_i : E \to \mathbf{R}_+,\ i = 1, \ldots, m$, where $m < n/4$

Output:

m edge disjoint Hamiltonian cycles $H_1, \ldots, H_m$

Time complexity:

$O(mn^2)$

Algorithm $\widetilde{A}$ for minimum-weight m-PSP

Input:

A complete n-vertex graph $G = (V, E)$ with weight functions $w_i : E \to \mathbf{R}_+,\ i = 1, \ldots, m$, where $m < n/4$

Output:

m edge disjoint Hamiltonian cycles $H_1, \ldots, H_m$

Time complexity:

$O(mn^2)$

Main idea:

modification of the greedy algorithm; finding each Hamiltonian cycle by turns.

Stage $i = 1, \ldots, m$.

In Stage i we consider given graph G with weight function w_i and construct Hamiltonian cycle H_i in 3 steps.

The description of Algorithm $\widetilde{A}$ for minimum-weight m-PSP

Stage $i = 1, \ldots, m$.

In Stage i we consider given graph G with weight function w_i and construct Hamiltonian cycle H_i in 3 steps.

Step i0

Choose the first vertex to start with, let it be vertex 1. Among all neighbors of 1 randomly choose a vertex v.

The description of Algorithm $\widetilde{A}$ for minimum-weight m-PSP

Stage $i = 1, \ldots, m$.

In Stage i we consider given graph G with weight function w_i and construct Hamiltonian cycle H_i in 3 steps.

Step i0

Choose the first vertex to start with, let it be vertex 1. Among all neighbors of 1 randomly choose a vertex v.

Step i1

Build a partial path applying the principle "go to the nearest not visited vertex" $n - 4i$ times, but do not use the vertex v in this Step.

The description of Algorithm $\widetilde{A}$ for minimum-weight m-PSP

Stage $i = 1, \ldots, m$.

In Stage i we consider given graph G with weight function w_i and construct Hamiltonian cycle H_i in 3 steps.

Step i0

Choose the first vertex to start with, let it be vertex 1. Among all neighbors of 1 randomly choose a vertex v.

Step i1

Build a partial path applying the principle "go to the nearest not visited vertex" $n - 4i$ times, but do not use the vertex v in this Step.

Step i2

This partial path is converted to a Hamiltonian cycle H_i via procedure $\mathbb{P}$.

Stage $i = 1, \ldots, m$.

In Stage i we consider given graph G with weight function w_i and construct Hamiltonian cycle H_i in 3 steps.

Step i0

Choose the first vertex to start with, let it be vertex 1. Among all neighbors of 1 randomly choose a vertex v.

Step i1

Build a partial path applying the principle "go to the nearest not visited vertex" $n - 4i$ times, but do not use the vertex v in this Step.

Step i2

This partial path is converted to a Hamiltonian cycle H_i via procedure $\mathbb{P}$.

For the formation of all further Hamiltonian cycles $i + 1, \ldots, m$ forbid all edges in H_i and the corresponding reverse edges.

The auxiliary procedure $\mathbb{P}$

Input:

The auxiliary procedure $\mathbb{P}$

Input:

1. given undirected $\hat{n}$-vertex graph $H(V_H, E_H)$ such that $\forall v \in V_H \quad deg(v) > \hat{n}/2$,
2. given vertices $u, v \in V_H$.

The auxiliary procedure $\mathbb{P}$

Input:

1. given undirected $\hat{n}$-vertex graph $H(V_H, E_H)$ such that $\forall v \in V_H \quad deg(v) > \hat{n}/2$,
2. given vertices $u, v \in V_H$.

Output:

Input:

1. given undirected $\hat{n}$-vertex graph $H(V_H, E_H)$ such that $\forall v \in V_H \quad deg(v) > \hat{n}/2$,
2. given vertices $u, v \in V_H$.

Output:

1. Hamiltonian path with endpoints u, v, or
2. Hamiltonian cycle if $u = v$.

The auxiliary procedure $\mathbb{P}$

Input:

1. given undirected $\hat{n}$-vertex graph $H(V_H, E_H)$ such that $\forall v \in V_H \;\; deg(v) > \hat{n}/2$,
2. given vertices $u, v \in V_H$.

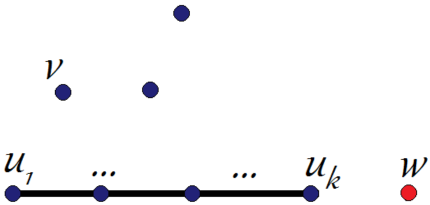
Output:

1. Hamiltonian path with endpoints u, v, or
2. Hamiltonian cycle if $u = v$.

Time complexity: $O(\hat{n}^2)$

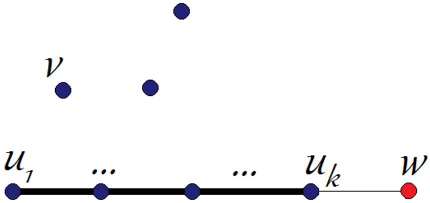
While $1 \leq k \leq \hat{n}$

- Let $P = \{u_1, \ldots, u_k\}$ be the constructed path.



While $1 \le k \le \hat{n}$

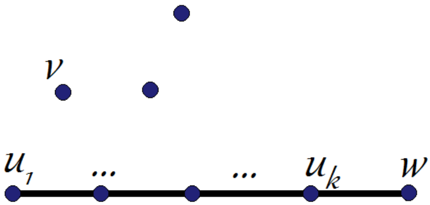
- Let $P = \{u_1, \dots, u_k\}$ be the constructed path.



- If there is an edge $\{u_k, w\} \in H$, where $w \notin P$, and $w = v \leftrightarrow k = \hat{n} - 1$, then

While $1 \le k \le \hat{n}$

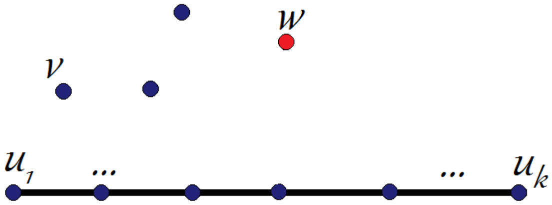
- Let $P = \{u_1, \dots, u_k\}$ be the constructed path.



- If there is an edge $\{u_k, w\} \in H$, where $w \notin P$, and $w = v \leftrightarrow k = \hat{n} - 1$, then
- $P := \{u = u_1, \dots, u_k, u_{k+1} = w\}$

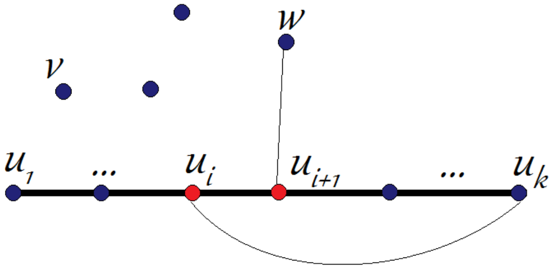
Otherwise:

- Randomly choose a vertex $w \notin P$ $(w = v \leftrightarrow k = \hat{n} - 1)$.



Otherwise:

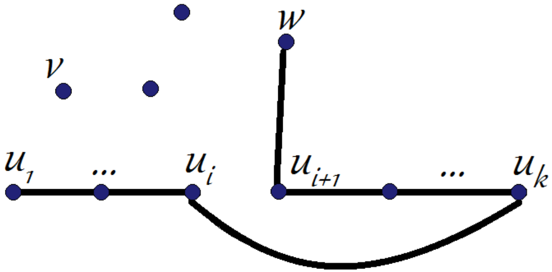
- Randomly choose a vertex $w \notin P$ ($w = v \leftrightarrow k = \hat{n} - 1$).



- Find edges $\{u_k, u_i\}$ and $\{w, u_{i+1}\}$

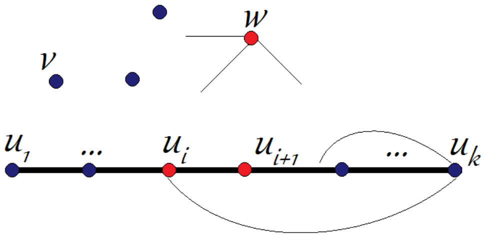
Otherwise:

- Randomly choose a vertex $w \notin P$ ($w = v \leftrightarrow k = \hat{n} - 1$).



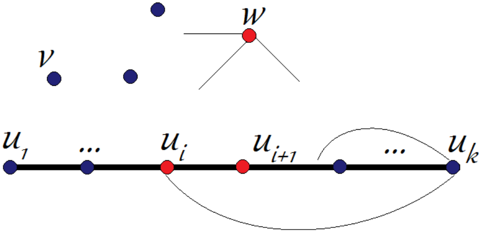
- Find edges $\{u_k, u_i\}$ and $\{w, u_{i+1}\}$
- $P := \{u = u_1, \ldots, u_i, u_k, \ldots, u_{i+1}, w\}$

Correctness of the procedure $\mathbb{P}$



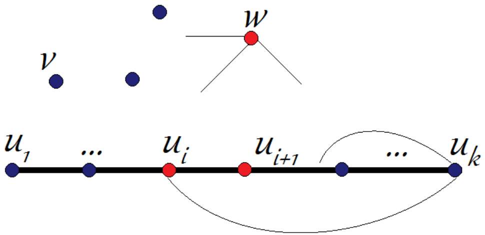
- Suppose, there is no edge $\{u_i, u_{i+1}\} \in P$ such that $\{u_k, u_i\}$ and $\{w, u_{i+1}\} \in E_H$.

Correctness of the procedure $\mathbb{P}$



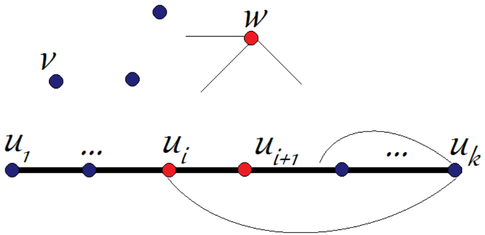
- Suppose, there is no edge $\{u_i, u_{i+1}\} \in P$ such that $\{u_k, u_i\}$ and $\{w, u_{i+1}\} \in E_H$.
- There are $> \hat{n}/2$ vertices adjacent to w.

Correctness of the procedure $\mathbb{P}$



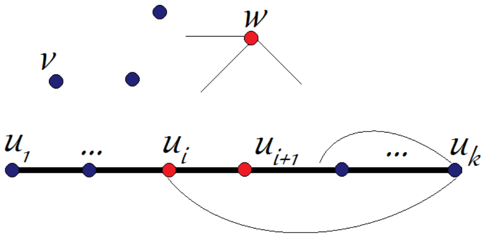
- Suppose, there is no edge $\{u_i, u_{i+1}\} \in P$ such that $\{u_k, u_i\}$ and $\{w, u_{i+1}\} \in E_H$.
- There are $> \hat{n}/2$ vertices adjacent to w.
- Vertices not adjacent to w: w, u_k, and u_{i+1}, where $i : \{u_k, u_i\} \in E_H$.

Correctness of the procedure $\mathbb{P}$



- Suppose, there is no edge $\{u_i, u_{i+1}\} \in P$ such that $\{u_k, u_i\}$ and $\{w, u_{i+1}\} \in E_H$.
- There are $> \hat{n}/2$ vertices adjacent to w.
- Vertices not adjacent to w: w, u_k, and u_{i+1}, where $i : \{u_k, u_i\} \in E_H$.
- So there are $> 1 + 1 + \hat{n}/2 - 2 = \hat{n}/2$ vertices that are not adjacent to w.

Correctness of the procedure $\mathbb{P}$



- Suppose, there is no edge $\{u_i, u_{i+1}\} \in P$ such that $\{u_k, u_i\}$ and $\{w, u_{i+1}\} \in E_H$.
- There are $> \hat{n}/2$ vertices adjacent to w.
- Vertices not adjacent to w: w, u_k, and u_{i+1}, where $i : \{u_k, u_i\} \in E_H$.
- So there are $> 1 + 1 + \hat{n}/2 - 2 = \hat{n}/2$ vertices that are not adjacent to w.

Contradiction.

Random inputs for the m-PSP

We represent an input for the m-PSP as a

$m \times n \times n$ cost matrix $C = \left(c_{ijk}\right)$, where c_{ijk} is equal to the i-th weight function $w_i(e)$ of edge $e = (j, k)$, $i = \overline{1, m}$,$j, k = \overline{1, n}$.

Random inputs for the m-PSP

We represent an input for the m-PSP as a

$m \times n \times n$ cost matrix $C = \left(c_{ijk}\right)$, where c_{ijk} is equal to the i-th weight function $w_i(e)$ of edge $e = (j, k)$, $i = \overline{1, m}, j, k = \overline{1, n}$.

Random input for the m-PSP is a

$m \times n \times n$ cost matrix $C = \left(c_{ijk}\right)$, which elements c_{ijk} are independent identically distributed random real numbers.

Random inputs for the m-PSP

Definition of distribution functions of majorizing type

The distribution function $\mathcal{F}'(x)$ is a function of $\mathcal{F}$-majorizing type if

$$\mathcal{F}'(x) \geq \mathcal{F}(x) \quad \text{for every x}$$

Random inputs for the m-PSP

Definition of distribution functions of majorizing type

The distribution function $\mathcal{F}'(x)$ is a function of $\mathcal{F}$-majorizing type if

$$\mathcal{F}'(x) \geq \mathcal{F}(x) \quad \text{for every x}$$

Distribution functions considered in the report

We will consider the random inputs for m-PSP with the following distribution functions

Random inputs for the m-PSP

Definition of distribution functions of majorizing type

The distribution function $\mathcal{F}'(x)$ is a function of $\mathcal{F}$-majorizing type if

$$\mathcal{F}'(x) \geq \mathcal{F}(x) \quad \text{for every x}$$

Distribution functions considered in the report

We will consider the random inputs for m-PSP with the following distribution functions

- of UNI$[a_n, b_n]$-majorizing type,
  where UNI$[a_n, b_n]$ is uniform distribution in the interval $[a_n, b_n]$, $0 < a_n < b_n$;

Random inputs for the m-PSP

Definition of distribution functions of majorizing type

The distribution function $\mathcal{F}'(x)$ is a function of $\mathcal{F}$-majorizing type if

$$\mathcal{F}'(x) \geq \mathcal{F}(x) \quad \text{for every x}$$

Distribution functions considered in the report

We will consider the random inputs for m-PSP with the following distribution functions

- of UNI$[a_n, b_n]$-majorizing type,
  where UNI$[a_n, b_n]$ is uniform distribution in the interval $[a_n, b_n]$, $0 < a_n < b_n$;
- of $\mathcal{F}_\beta$-majorizing type,
  where $\mathcal{F}_\beta(x)$ is exponential distribution with parameter $\beta = \beta_n$:
  $$\mathcal{F}_\beta(x) = 1 - \exp\left(\frac{x - a_n}{\beta}\right), \; x \geq a_n > 0.$$

An algorithm A has performance bounds $\varepsilon_A(n), \delta_A(n)$ if

$$\mathbf{Pr}\Big\{F_A > \big(1+\varepsilon_A(n)\big)\,OPT\Big\} \le \delta_A(n)$$

Definitions

An algorithm A has performance bounds $\varepsilon_A(n), \delta_A(n)$ if

$$\mathbf{Pr}\Big\{F_A > \big(1 + \varepsilon_A(n)\big)\,OPT\Big\} \leq \delta_A(n)$$

- n is the problem size

An algorithm A has performance bounds $\varepsilon_A(n), \delta_A(n)$ if

$$\mathbf{Pr}\Big\{F_A > \big(1+\varepsilon_A(n)\big)\,OPT\Big\} \le \delta_A(n)$$

- n is the problem size
- $\varepsilon_A(n)$ is the relative error of the algorithm

Definitions

An algorithm A has performance bounds $\varepsilon_A(n), \delta_A(n)$ if

$$\mathbf{Pr}\Big\{F_A > \big(1 + \varepsilon_A(n)\big)\,OPT\Big\} \leq \delta_A(n)$$

- n is the problem size
- $\varepsilon_A(n)$ is the relative error of the algorithm
- $\delta_A(n)$ is the failure probability of the algorithm, i.e. the proportion of cases when algorithm A doesn't hold the relative error $\varepsilon_A(n)$.

An algorithm A has performance bounds $\varepsilon_A(n), \delta_A(n)$ if

$$\mathbf{Pr}\Big\{F_A > \big(1+\varepsilon_A(n)\big)OPT\Big\} \leq \delta_A(n)$$

- n is the problem size
- $\varepsilon_A(n)$ is the relative error of the algorithm
- $\delta_A(n)$ is the failure probability of the algorithm, i.e. the proportion of cases when algorithm A doesn't hold the relative error $\varepsilon_A(n)$.

An algorithm A is called asymptotically optimal (exact)

on a class of instances, if there exist performance bounds s.t.

$$\varepsilon_A(n) \xrightarrow[n\to\infty]{} 0, \quad \delta_A(n) \xrightarrow[n\to\infty]{} 0.$$

Let $H_i = \{e_1^{(i)}, \ldots, e_n^{(i)}\}$ is i-th constructed Hamiltonian cycle

Probabilistic analysis of Algorithm $\widetilde{A}$

Let $H_i = \{e_1^{(i)}, \ldots, e_n^{(i)}\}$ is i-th constructed Hamiltonian cycle

Performance guarantees for Algorithm $\widetilde{A}$ will be defined by the inequality:

$$Pr\Big\{ \sum_{i=1}^{m} \sum_{s=1}^{n} w_i(e_s^{(i)}) > (1 + \varepsilon_{\widetilde{A}}) OPT \Big\} \leq \delta_{\widetilde{A}}.$$

Probabilistic analysis of Algorithm $\widetilde{A}$

Let $H_i = \{e_1^{(i)}, \ldots, e_n^{(i)}\}$ is i-th constructed Hamiltonian cycle

Performance guarantees for Algorithm $\widetilde{A}$ will be defined by the inequality:

$$Pr\Big\{ \sum_{i=1}^{m} \sum_{s=1}^{n} w_i(e_s^{(i)}) > (1+\varepsilon_{\widetilde{A}})OPT \Big\} \le \delta_{\widetilde{A}}.$$

Denote $\xi_{is} = w_i(e_s^{(i)})$. Then

$$Pr\Big\{ \sum_{i=1}^{m} \sum_{s=1}^{n} \xi_{is} > (1+\varepsilon_{\widetilde{A}})OPT \Big\} \le \delta_{\widetilde{A}}.$$

Main ideas of the probabilistic analysis

Main ideas of the probabilistic analysis

- All ξ_{is} are independent random variables.

Main ideas of the probabilistic analysis

- All ξ_{is} are independent random variables.
- Weight ξ_{is} of an edge chosen in Step 1(greedy algorithm) is estimated from above as minimum of $n - 2i - s + 2$ elements of random input.

Main ideas of the probabilistic analysis

- All ξ_{is} are independent random variables.
- Weight ξ_{is} of an edge chosen in Step 1(greedy algorithm) is estimated from above as minimum of $n - 2i - s + 2$ elements of random input.
- Weight ξ_{is} of an edge chosen in Step 2(procedure $\mathbb{P}$) has the same distribution function as the elements of random input.

Main ideas of the probabilistic analysis

- All ξ_{is} are independent random variables.
- Weight ξ_{is} of an edge chosen in Step 1(greedy algorithm) is estimated from above as minimum of $n - 2i - s + 2$ elements of random input.
- Weight ξ_{is} of an edge chosen in Step 2(procedure $\mathbb{P}$) has the same distribution function as the elements of random input.
- Use the inequality $OPT \geq a_n mn$

Main ideas of the probabilistic analysis

- All ξ_{is} are independent random variables.
- Weight ξ_{is} of an edge chosen in Step 1(greedy algorithm) is estimated from above as minimum of $n-2i-s+2$ elements of random input.
- Weight ξ_{is} of an edge chosen in Step 2(procedure $\mathbb{P}$) has the same distribution function as the elements of random input.
- Use the inequality $OPT \geq a_n mn$
- Use Petrov's Theorem

Petrov V.V. 'Limit theorems for sums of independent random variables', 1987

Petrov's Theorem

Consider independent random variables $\eta_1, \ldots, \eta_n$ and $S = \Sigma_{k=1}^{n} \eta_k$. Let there be positive constants $g_1, \ldots, g_n$ and T, such that

$$\mathbf{E} e^{t\eta_k} \leq e^{\frac{g_k t^2}{2}},\ 0 \leq t \leq T,\ k = 1, \ldots, n$$

Denote $\mathcal{G} = \Sigma_{k=1}^{n} g_k$. Then

$$\mathbf{Pr}\{S \geq x\} \leq \begin{cases} e^{\frac{-x^2}{2G}}, & 0 \leq x \leq \mathcal{G}T, \\ e^{\frac{-Tx}{2}}, & x \geq \mathcal{G}T \end{cases}$$

Where $\mathbf{E}X$ is the expected value of random variable X.

To apply Petrov's theorem we used the previous results from the following papers to obtain constants g_{is} for the random variables ξ_{is}

To apply Petrov's theorem we used the previous results from the following papers to obtain constants g_{is} for the random variables ξ_{is}

For uniform distribution function:

E. Kh. Gimadi,Yu. V. Glazkov *An asymptotically exact algorithm for one modification of planar three-index assignment problem//* Journal of Applied and Industrial Mathematics December 2007, Volume 1, Issue 4, pp 442-452

Probabilistic analysis of Algorithm $\widetilde{A}$

To apply Petrov's theorem we used the previous results from the following papers to obtain constants g_{is} for the random variables ξ_{is}

For uniform distribution function:

E. Kh. Gimadi,Yu. V. Glazkov *An asymptotically exact algorithm for one modification of planar three-index assignment problem*// Journal of Applied and Industrial Mathematics December 2007, Volume 1, Issue 4, pp 442-452

For exponential distribution function:

E. Kh. Gimadi, A. Le Gallou, A. V. Shakhshneyder, *Probabilistic analysis of an approximation algorithm for the traveling salesman problem on unbounded above instances*// Journal of Applied and Industrial Mathematics April 2009, Volume 3, Issue 2, pp 207-221.

The performance bounds of the algorithm obtained for random inputs of m-PSP with some distribution function $F(x)$ will also be true for random inputs with any distribution function of $F(x)$-majorizing type.

Statement 1

Let $\xi_1, ..., \xi_k$ be the independent random variables with distribution function $F(x)$,
Let $\hat{F}(x)$ be the distribution function of $\xi = \min(\xi_1, ..., \xi_k)$,
Let $\eta_1, ..., \eta_k$ be the independent random variables with distribution function $G(x)$,
Let $\hat{G}(x)$ be the distribution function of $\eta = \min(\eta_1, ..., \eta_k)$.

Then for any x

$$F(x) \le G(x) \Rightarrow \hat{F}(x) \le \hat{G}(x).$$

Distribution functions of majorizing type

The performance bounds of the algorithm obtained for random inputs of m-PSP with some distribution function $F(x)$ will also be true for random inputs with any distribution function of $F(x)$-majorizing type.

Statement 1

Let $\xi_1, ..., \xi_k$ be the independent random variables with distribution function $F(x)$,
Let $\hat{F}(x)$ be the distribution function of $\xi = \min(\xi_1, ..., \xi_k)$,
Let $\eta_1, ..., \eta_k$ be the independent random variables with distribution function $G(x)$,
Let $\hat{G}(x)$ be the distribution function of $\eta = \min(\eta_1, ..., \eta_k)$.

Then for any x

$$F(x) \le G(x) \Rightarrow \hat{F}(x) \le \hat{G}(x).$$

Distribution functions of majorizing type

The performance bounds of the algorithm obtained for random inputs of m-PSP with some distribution function $F(x)$ will also be true for random inputs with any distribution function of $F(x)$-majorizing type.

Statement 1

Let $\xi_1, ..., \xi_k$ be the independent random variables with distribution function $F(x)$,
Let $\hat{F}(x)$ be the distribution function of $\xi = \min(\xi_1, ..., \xi_k)$,
Let $\eta_1, ..., \eta_k$ be the independent random variables with distribution function $G(x)$,
Let $\hat{G}(x)$ be the distribution function of $\eta = \min(\eta_1, ..., \eta_k)$.

Then for any x

$$F(x) \leq G(x) \Rightarrow \hat{F}(x) \leq \hat{G}(x).$$

The statement follows directly from the equations

$$\hat{F}(x) = 1 - (1 - F(x))^k \quad \text{and} \quad \hat{G}(x) = 1 - (1 - G(x))^k.$$

Distribution functions of majorizing type

Statement 2

Let $P_\xi, P_\eta, P_\zeta, P_\chi$ be the distribution functions of random variables ξ, η, ζ, χ, respectively. And let ξ and ζ be independent, η and χ be independent. Then

$$(\forall x\ P_\xi(x) \leq P_\eta(x)) \wedge (\forall y\ P_\zeta(y) \leq P_\chi(y)) \Rightarrow (\forall z\ P_{\xi+\zeta}(z) \leq P_{\eta+\chi}(z)).$$

Proof

$$P_{\xi+\zeta}(x) = \int\limits_{-\infty}^{\infty} P_\xi(x-y)dP_\zeta(y) \leq \int\limits_{-\infty}^{\infty} P_\eta(x-y)dP_\zeta(y)$$

$$= P_{\eta+\zeta}(x) = \int\limits_{-\infty}^{\infty} P_\zeta(x-y)dP_\eta(y) \leq \int\limits_{-\infty}^{\infty} P_\chi(x-y)dP_\eta(y) = P_{\eta+\chi}(x).$$

Distribution functions of majorizing type

Theorem

Let the distribution function $F(x)$ of random inputs of m-PSP be s.t.

$$F(x) \geq P(x).$$

Then Algorithm $\widetilde{A}$ has the same performance guarantees $(\varepsilon_{\widetilde{A}}, \delta_{\widetilde{A}})$ on these random inputs, as it would have on random inputs with distribution function $P(x)$.

Theorem

Let the distribution function $F(x)$ of random inputs of m-PSP be s.t.

$$F(x) \geq P(x).$$

Then Algorithm $\widetilde{A}$ has the same performance guarantees $(\varepsilon_{\widetilde{A}}, \delta_{\widetilde{A}})$ on these random inputs, as it would have on random inputs with distribution function $P(x)$.

The proof follows from Statements 1-2, and the fact that all the weights of all edges that belong to the constructed solution of m-PSP are independent random variables.

Distribution functions of majorizing type

Theorem

Let the distribution function $F(x)$ of random inputs of m-PSP be s.t.

$$F(x) \geq P(x).$$

Then Algorithm $\widetilde{A}$ has the same performance guarantees $(\varepsilon_{\widetilde{A}}, \delta_{\widetilde{A}})$ on these random inputs, as it would have on random inputs with distribution function $P(x)$.

The proof follows from Statements 1-2, and the fact that all the weights of all edges that belong to the constructed solution of m-PSP are independent random variables.

Corollary (for example)

The performance guarantees of Algorithm $\widetilde{A}$ obtained in the case of random inputs with **exponential** distribution with a parameter β will also hold in case of random inputs with **truncated normal** distribution function with a certain parameter σ_n.

The conditions of the asymptotic optimality of Algorithm $\widetilde{A}$

For the random inputs of m-PSP with the distribution function of **UNI**$[a_n, b_n]$-**majorizing type**, $0 < a_n < b_n$, Algorithm $\widetilde{A}$ is asymptotically exact with the following performance guarantees

The conditions of the asymptotic optimality of Algorithm $\widetilde{A}$

For the random inputs of m-PSP with the distribution function of **UNI**$[a_n, b_n]$**-majorizing type**, $0 < a_n < b_n$, Algorithm $\widetilde{A}$ is asymptotically exact with the following performance guarantees

for $2 \le m \le \ln n$

$$\varepsilon_{\widetilde{A}} = O\left(\frac{b_n/a_n}{n/\ln n}\right), \quad \delta_{\widetilde{A}} = n^{-9},$$

The conditions of the asymptotic optimality of Algorithm $\widetilde{A}$

For the random inputs of m-PSP with the distribution function of **UNI**$[a_n, b_n]$-**majorizing type**, $0 < a_n < b_n$, Algorithm $\widetilde{A}$ is asymptotically exact with the following performance guarantees

for $2 \le m \le \ln n$

$$\varepsilon_{\widetilde{A}} = O\left(\frac{b_n/a_n}{n/\ln n}\right), \quad \delta_{\widetilde{A}} = n^{-9},$$

$$\text{if} \quad \frac{b_n}{a_n} = o\left(\frac{n}{\ln n}\right);$$

The conditions of the asymptotic optimality of Algorithm $\widetilde{A}$

For the random inputs of m-PSP with the distribution function of **UNI**$[a_n, b_n]$-**majorizing type**, $0 < a_n < b_n$, Algorithm $\widetilde{A}$ is asymptotically exact with the following performance guarantees

for $2 \le m \le \ln n$

$$\varepsilon_{\widetilde{A}} = O\left(\frac{b_n/a_n}{n/\ln n}\right), \quad \delta_{\widetilde{A}} = n^{-9},$$

$$\text{if } \frac{b_n}{a_n} = o\left(\frac{n}{\ln n}\right);$$

for $\ln n < m \le n^{1-\theta} < n/4$

$$\varepsilon_{\widetilde{A}} = O\left(\frac{b_n/a_n}{n^{\theta}}\right), \quad \delta_{\widetilde{A}} = n^{-9},$$

The conditions of the asymptotic optimality of Algorithm $\widetilde{A}$

For the random inputs of m-PSP with the distribution function of **UNI**$[a_n, b_n]$**-majorizing type**, $0 < a_n < b_n$, Algorithm $\widetilde{A}$ is asymptotically exact with the following performance guarantees

for $2 \le m \le \ln n$

$$\varepsilon_{\widetilde{A}} = O\left(\frac{b_n/a_n}{n/\ln n}\right), \quad \delta_{\widetilde{A}} = n^{-9},$$

$$\text{if} \quad \frac{b_n}{a_n} = o\left(\frac{n}{\ln n}\right);$$

for $\ln n < m \le n^{1-\theta} < n/4$

$$\varepsilon_{\widetilde{A}} = O\left(\frac{b_n/a_n}{n^\theta}\right), \quad \delta_{\widetilde{A}} = n^{-9},$$

$$\text{if} \quad \frac{b_n}{a_n} = o(n^\theta).$$

The conditions of the asymptotic optimality of Algorithm $\widetilde{A}$

For the random inputs of m-PSP with the distribution function of **exponential** $\mathcal{F}_\beta$**-majorizing type**, Algorithm $\widetilde{A}$ is asymptotically exact with the following performance guarantees

The conditions of the asymptotic optimality of Algorithm $\widetilde{A}$

For the random inputs of m-PSP with the distribution function of **exponential** $\mathcal{F}_\beta$**-majorizing type**, Algorithm $\widetilde{A}$ is asymptotically exact with the following performance guarantees

for $2 \le m \le \ln n$

$$\varepsilon_{\widetilde{A}} = O\left(\frac{\beta/a_n}{n/\ln n}\right), \quad \delta_{\widetilde{A}} = n^{-3m/4},$$

The conditions of the asymptotic optimality of Algorithm $\widetilde{A}$

For the random inputs of m-PSP with the distribution function of **exponential $\mathcal{F}_\beta$-majorizing type**, Algorithm $\widetilde{A}$ is asymptotically exact with the following performance guarantees

for $2 \leq m \leq \ln n$

$$\varepsilon_{\widetilde{A}} = O\left(\frac{\beta / a_n}{n / \ln n}\right), \quad \delta_{\widetilde{A}} = n^{-3m/4},$$

$$\text{if } \frac{b_n}{a_n} = o\left(\frac{n}{\ln n}\right);$$

The conditions of the asymptotic optimality of Algorithm $\widetilde{A}$

For the random inputs of m-PSP with the distribution function of **exponential $\mathcal{F}_\beta$-majorizing type**, Algorithm $\widetilde{A}$ is asymptotically exact with the following performance guarantees

for $2 \le m \le \ln n$

$$\varepsilon_{\widetilde{A}} = O\left(\frac{\beta/a_n}{n/\ln n}\right), \quad \delta_{\widetilde{A}} = n^{-3m/4},$$

$$\text{if } \frac{b_n}{a_n} = o\left(\frac{n}{\ln n}\right);$$

for $\ln n < m \le n^{1-\theta} < n/4$

$$\varepsilon_{\widetilde{A}} = O\left(\frac{\beta/a_n}{n^\theta}\right), \quad \delta_{\widetilde{A}} = n^{-3m/4},$$

The conditions of the asymptotic optimality of Algorithm $\widetilde{A}$

For the random inputs of m-PSP with the distribution function of **exponential $\mathcal{F}_\beta$-majorizing type**, Algorithm $\widetilde{A}$ is asymptotically exact with the following performance guarantees

for $2 \le m \le \ln n$

$$\varepsilon_{\widetilde{A}} = O\left(\frac{\beta/a_n}{n/\ln n}\right), \quad \delta_{\widetilde{A}} = n^{-3m/4},$$

$$\text{if} \quad \frac{b_n}{a_n} = o\left(\frac{n}{\ln n}\right);$$

for $\ln n < m \le n^{1-\theta} < n/4$

$$\varepsilon_{\widetilde{A}} = O\left(\frac{\beta/a_n}{n^\theta}\right), \quad \delta_{\widetilde{A}} = n^{-3m/4},$$

$$\text{if} \quad \frac{b_n}{a_n} = o(n^\theta).$$

Last publication of authors on the theme of the report

The paper

Э.Х. Гимади, А.М. Истомин, И.А. Рыков, О.Ю. Цидулко. Вероятностный анализ приближённого алгоритма для решения задачи нескольких коммивояжеров на случайных входных данных, неограниченных сверху // Труды ИММ УрО РАН. 2014. Т. 20, № 2, С. 88-98.
Probabilistic analysis of an approximation algorithm for the m-peripatetic salesman problem on random instances unbounded from above.

Thank you!

Thank you for your attention!

Algorithm $\widetilde{A}$ solving the m-PSP

- **Input:** A complete n-vertex graph $G = (V, E)$ with weight functions $w_i : E \to \mathbf{R}_+,\ i = 1, \ldots, m$, where $m < n/4$
- **Output:** m edge disjoint Hamiltonian cycles $H_1, \ldots, H_m$
- **Time complexity:** $O(mn^2)$

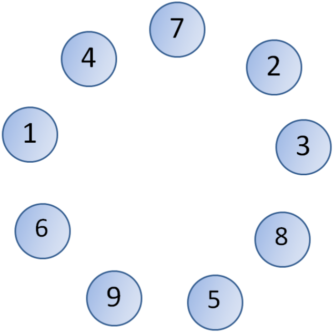
Algorithm $\widetilde{A}$ solving the m-PSP

- **Input:** A complete n-vertex graph $G = (V, E)$ with weight functions $w_i : E \to \mathbf{R}_+,\ i = 1, \ldots, m$, where $m < n/4$
- **Output:** m edge disjoint Hamiltonian cycles $H_1, \ldots, H_m$
- **Time complexity:** $O(mn^2)$
- **Main idea:** modification of the greedy algorithm; finding each Hamiltonian cycle by turns.

Step 0

i – number of current Hamiltonian cycle.
F – set of forbidden edges (at first $F = \emptyset$).

1. Consider the traveling salesman problem for graph $G \setminus F$ with weight function w_i.

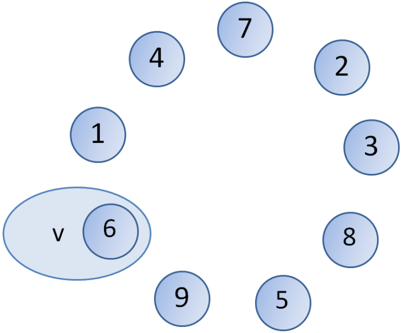


2. Randomly choose the first vertex to start with. Let it be vertex 1.
3. Among all neighbors of 1 randomly choose a vertex v.

Step 0

i – number of current Hamiltonian cycle.
F – set of forbidden edges (at first $F = \emptyset$).

1. Consider the traveling salesman problem for graph $G \setminus F$ with weight function w_i.

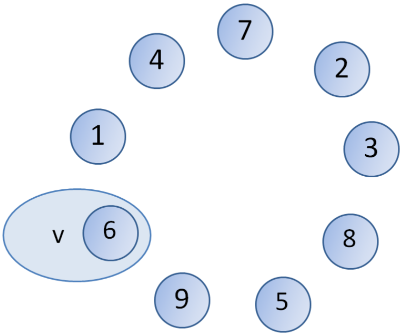


2. Randomly choose the first vertex to start with. Let it be vertex 1.
3. Among all neighbors of 1 randomly choose a vertex v.

Step 1

i – number of current Hamiltonian cycle.
s – number of processed vertices.

1. While $s < n - 4i$



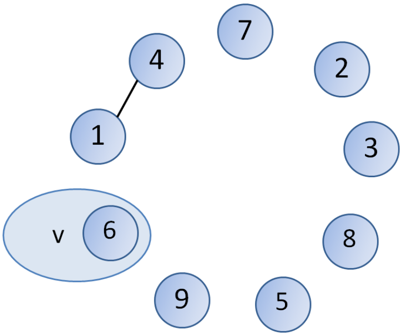
$i = 1, \ \ s = 1.$

2. "go to the nearest unvisited vertex, except vertex v.
3. $s := s + 1$.

Step 1

i – number of current Hamiltonian cycle.
s – number of processed vertices.

1. While $s < n - 4i$



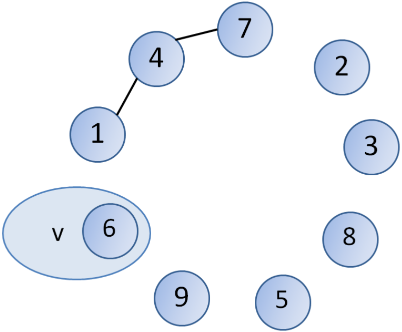
$$i = 1, \; s = 2.$$

2. "go to the nearest unvisited vertex, except vertex v.
3. $s := s + 1$.

Step 1

i – number of current Hamiltonian cycle.
s – number of processed vertices.

1. While $s < n - 4i$



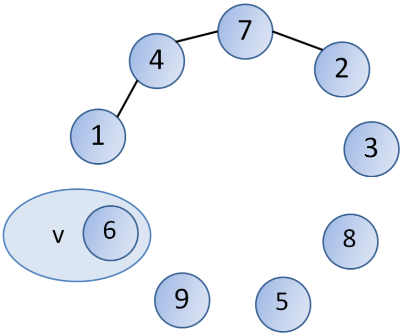
$i = 1, \;\; s = 3.$

2. "go to the nearest unvisited vertex, except vertex v.
3. $s := s + 1.$

Step 1

i – number of current Hamiltonian cycle.
s – number of processed vertices.

1. While $s < n - 4i$



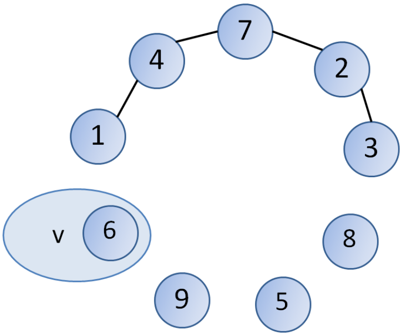
$i = 1, \; s = 4.$

2. "go to the nearest unvisited vertex, except vertex v.
3. $s := s + 1.$

Step 1

i – number of current Hamiltonian cycle.
s – number of processed vertices.

1. While $s < n - 4i$

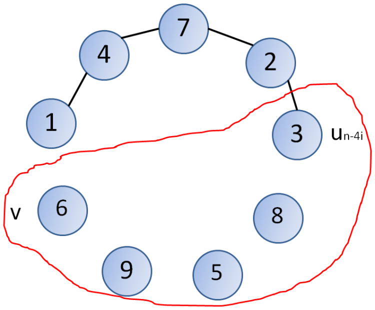


$i = 1, \ s = 5.$

2. "go to the nearest unvisited vertex, except vertex v.
3. $s := s + 1.$

Step 2

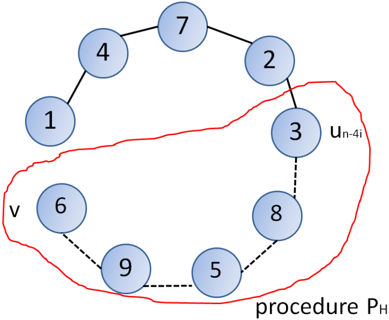
- Consider a subgraph H induced by all unprocessed vertices, and the last processed vertex:



- Using procedure $\mathbb{P}$ build a path with endpoints u_{n-4i}, v,
- Complete the Hamiltonian cycle H_i.
- For further stages forbid all edges $\in H_i$ and the corresponding reverse edges.

Step 2

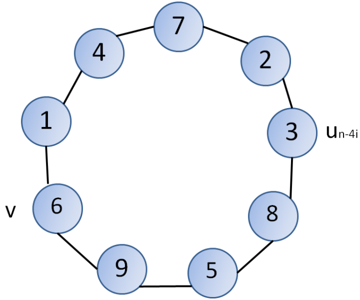
- Consider a subgraph H induced by all unprocessed vertices, and the last processed vertex:



- Using procedure $\mathbb{P}$ build a path with endpoints u_{n-4i}, v,
- Complete the Hamiltonian cycle H_i.
- For further stages forbid all edges $\in H_i$ and the corresponding reverse edges.

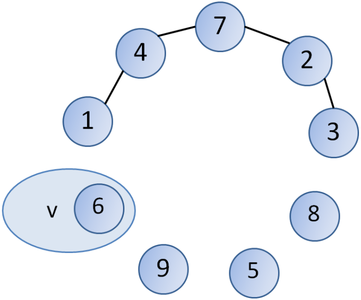
Step 2

- Consider a subgraph H induced by all unprocessed vertices, and the last processed vertex:

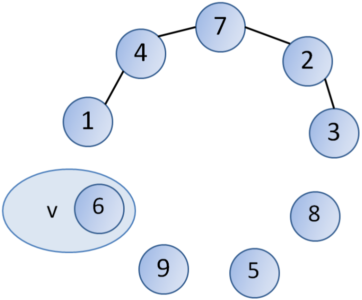


- Using procedure $\mathbb{P}$ build a path with endpoints u_{n-4i}, v,
- Complete the Hamiltonian cycle H_i.
- For further stages forbid all edges $\in H_i$ and the corresponding reverse edges.

In Step 1.



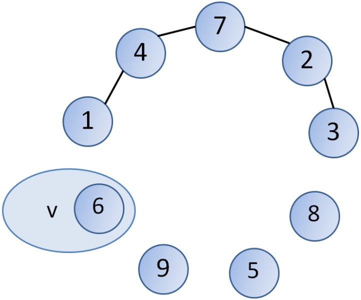
In Step 1.



- the degree of each vertex at the beginning of Step 1: $deg(v) = n - 2 - 2(i - 1) = n - 2i$

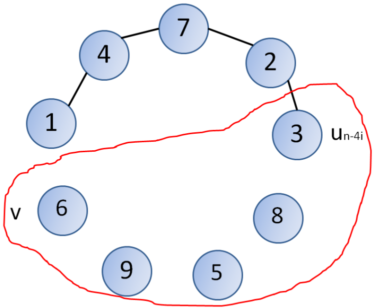
Relevance of Algorithm $\widetilde{A}$

In Step 1.

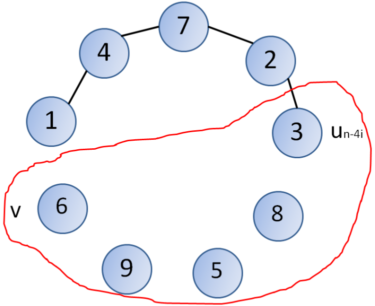


- the degree of each vertex at the beginning of Step 1: $deg(v) = n - 2 - 2(i - 1) = n - 2i$
- the greedy algorithm makes $n - 4i$ steps, so it is always possible to make the next step.

Consider subgraph H constructed in Step 2.



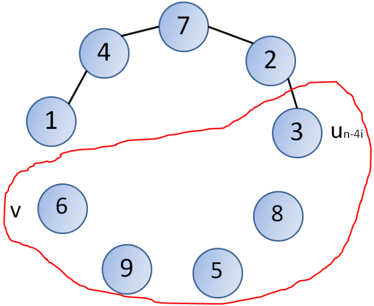
Consider subgraph H constructed in Step 2.



- Since $s = n - 4i$, $|V_H| = n - s + 1 = 4i + 1$.

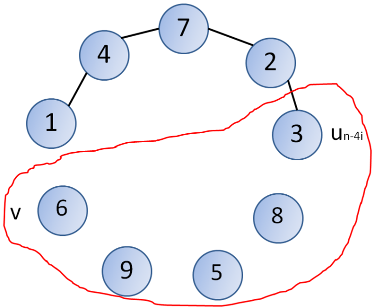
Relevance of Algorithm $\widetilde{A}$

Consider subgraph H constructed in Step 2.



- Since $s = n - 4i$, $|V_H| = n - s + 1 = 4i + 1$.
- $\forall v \in V_H \; deg(v) \geq (4i + 1 - 1) - 2(i - 1) = 2i + 2$

Consider subgraph H constructed in Step 2.



- Since $s = n - 4i$, $|V_H| = n - s + 1 = 4i + 1$.
- $\forall v \in V_H \; deg(v) \geq (4i + 1 - 1) - 2(i - 1) = 2i + 2$
- Thus we can use procedure $\mathbb{P}$ for this graph.

For each Hamiltonian cycle $H_1, \ldots, H_m$ we have:

Time complexity of Algorithm $\widetilde{A}$

For each Hamiltonian cycle $H_1, \ldots, H_m$ we have:

- Step 1 (greedy algorithm) – $O(n^2)$

Time complexity of Algorithm $\widetilde{A}$

For each Hamiltonian cycle $H_1, \ldots, H_m$ we have:

- Step 1 (greedy algorithm) – $O(n^2)$
- Step 2 (procedure P_H) – $O(n^2)$

Time complexity of Algorithm $\widetilde{A}$

For each Hamiltonian cycle $H_1, \ldots, H_m$ we have:

- Step 1 (greedy algorithm) – $O(n^2)$
- Step 2 (procedure P_H) – $O(n^2)$

Total time complexity: $O(mn^2)$.