Single approximation for Multiobjective Max TSP

Cristina Bazgan[1], Laurent Gourvès[1], Jérôme Monnot[1], Fanny Pascual[2]

[1] : LAMSADE, University of Paris Dauphine, France
[2] : LIP6, University Pierre et Marie Curie, France

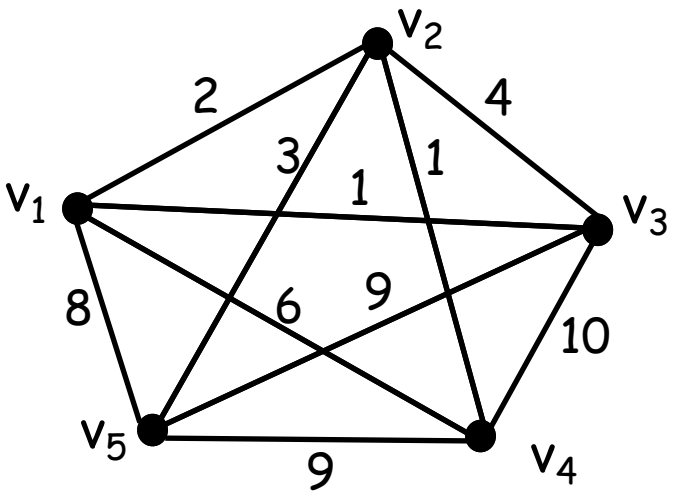
MCDM, June 16th 2011

Outline

- Introduction :
  - the Multiobjective Max TSP
  - our approach
- Results for the Biobjective Max TSP
  - lower bounds
  - a generic algorithm
  - analysis of one case
- Dealing with many objectives
- Conclusion and further research

Max Traveling Salesman Problem

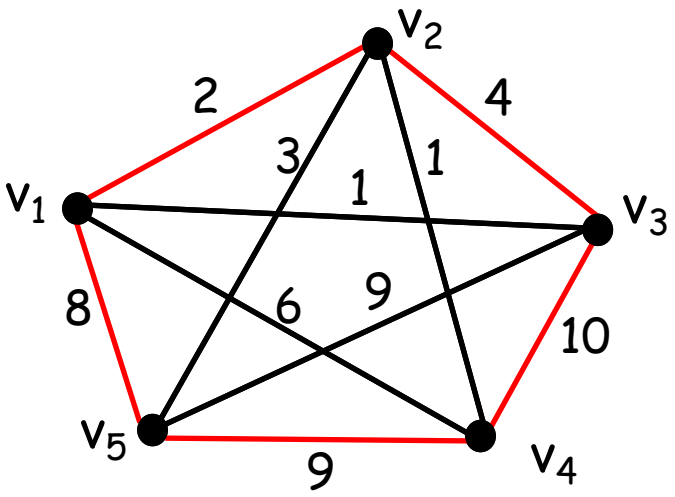
- Data: an undirected complete graph G=(V,E) with nonnegative weights on the edges. Let n=|V|.
- Return: an hamiltonian cycle of maximum weight (sum of its edges' weights)



NP-hard problem

Max Traveling Salesman Problem

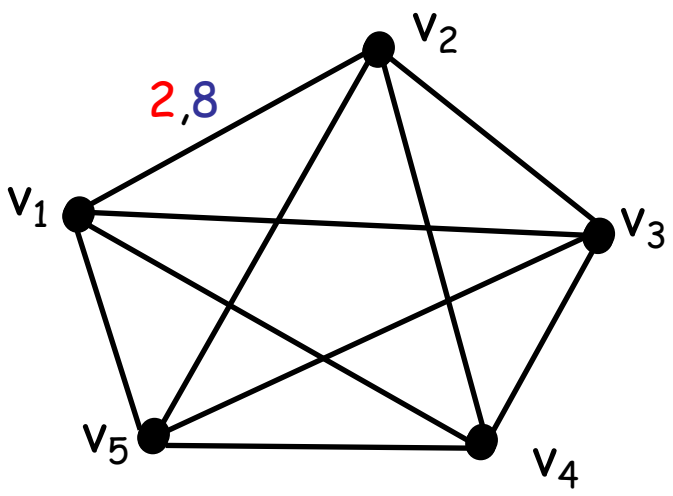
- Data: an undirected complete graph G=(V,E) with nonnegative weights on the edges. Let n=|V|.
- Return: an hamiltonian cycle of maximum weight (sum of its edges' weights)



NP-hard problem

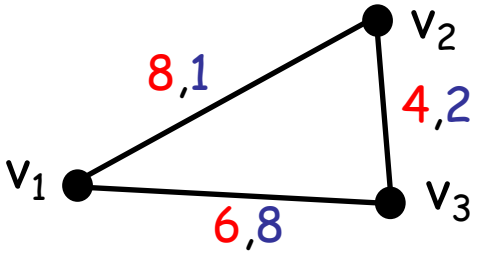
The Biobjective Max TSP

- Each edge has two weights : a and b.



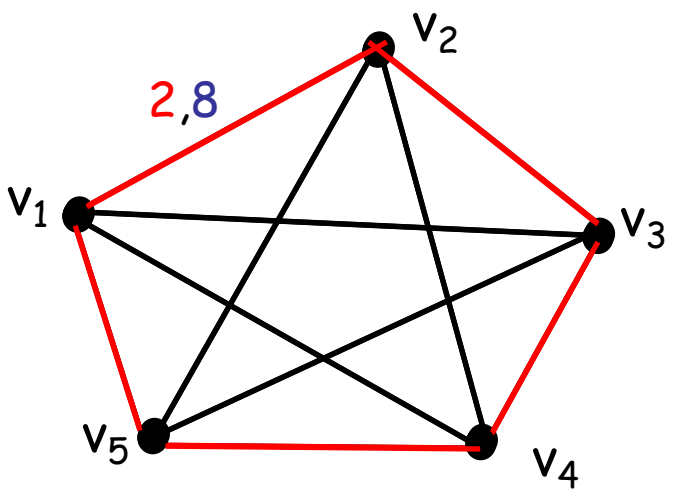
$a(v_1,v_2)$ = weight of the objective a on edge $\{v_1,v_2\}$.
$a(T)$ = weight of objective a on the set of edges T.

Triangle inequality :
$\forall (v_1,v_2,v_3) \in V^3,\ a(v_1,v_2) + a(v_2,v_3) \geq a(v_1,v_3)$



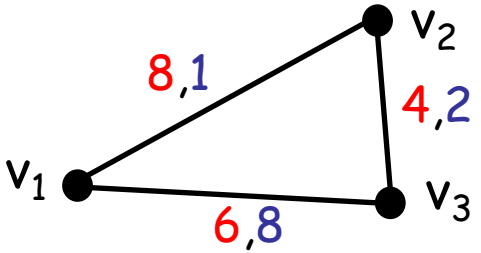
The Biobjective Max TSP

- Each edge has two weights : a and b.



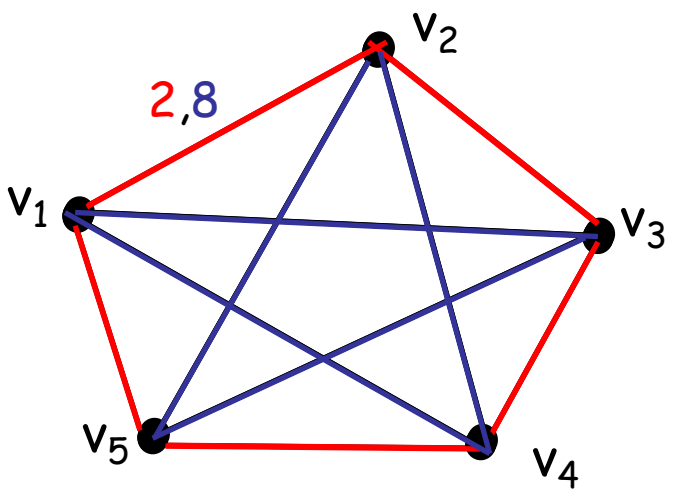
$a(v_1,v_2)$ = weight of the objective a on edge $\{v_1,v_2\}$.
$a(T)$ = weight of objective a on the set of edges T.

Triangle inequality :
$\forall (v_1,v_2,v_3) \in V^3,\ a(v_1,v_2) + a(v_2,v_3) \geq a(v_1,v_3)$



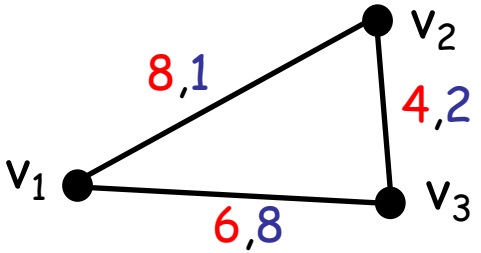
The Biobjective Max TSP

- Each edge has two weights : a and b.



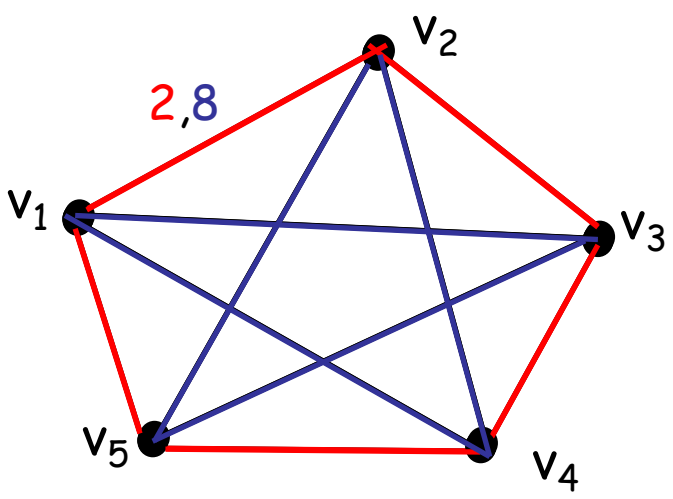
$a(v_1,v_2)$ = weight of the objective a on edge $\{v_1,v_2\}$.
$a(T)$ = weight of objective a on the set of edges T.

Triangle inequality :
$\forall (v_1,v_2,v_3) \in V^3,\ a(v_1,v_2) + a(v_2,v_3) \geq a(v_1,v_3)$



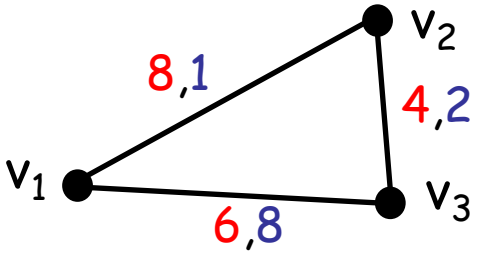
The Biobjective Max TSP

- Each edge has two weights : a and b.



$a(v_1,v_2)$ = weight of the objective a on edge $\{v_1,v_2\}$.
$a(T)$ = weight of objective a on the set of edges T.

Triangle inequality :
$\forall (v_1,v_2,v_3) \in V^3,\ a(v_1,v_2) + a(v_2,v_3) \geq a(v_1,v_3)$



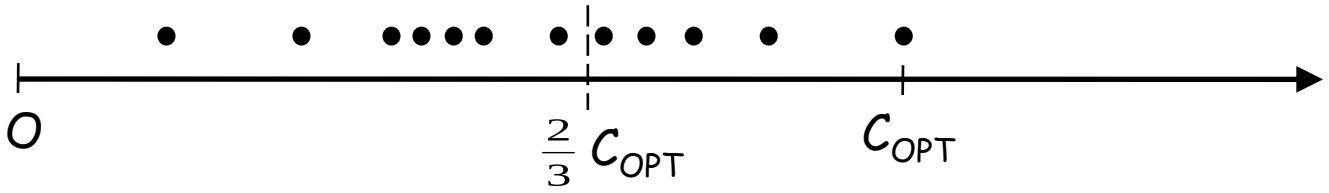
- Aim : output a good cycle for both objectives.
- 3 cases :

  0/1/2 objectives fulfill the triangle inequality

Approximation of the Max TSP

One objective: an algorithm is r-approximate ($r \leq 1$) iff

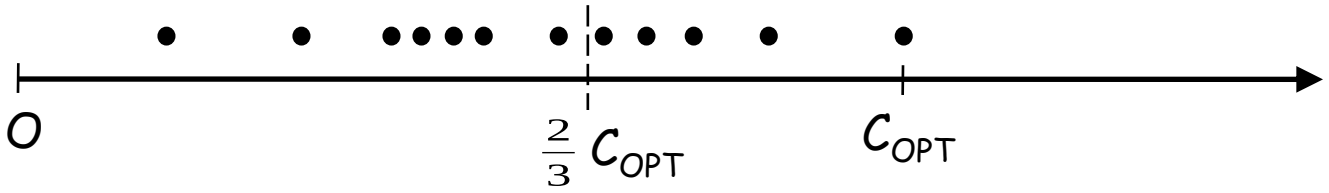
Cost (solution returned) $\geq$ r x Cost(optimal solution)



Approximation of the Max TSP

One objective: an algorithm is r-approximate ($r \leq 1$) iff

Cost (solution returned) $\geq$ r x Cost(optimal solution)

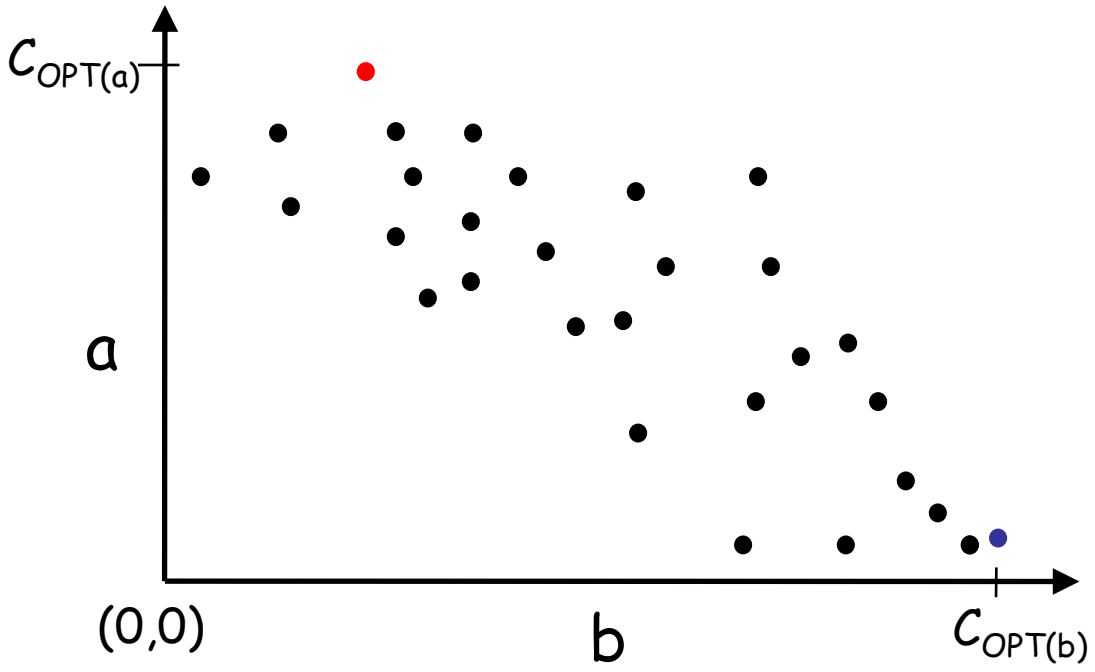


Best approximation ratio for the monobjective Max TSP:

- without the triangle inequality: $61/83 \cong 0.73$
  [Chen et al, Inf. Process. Letters, 1995]
- with the triangle inequality: $7/8 = 0.875$
  [Kowalik and Mucha, Theoretical Computer Science, 2009]

Approximation of the Max TSP

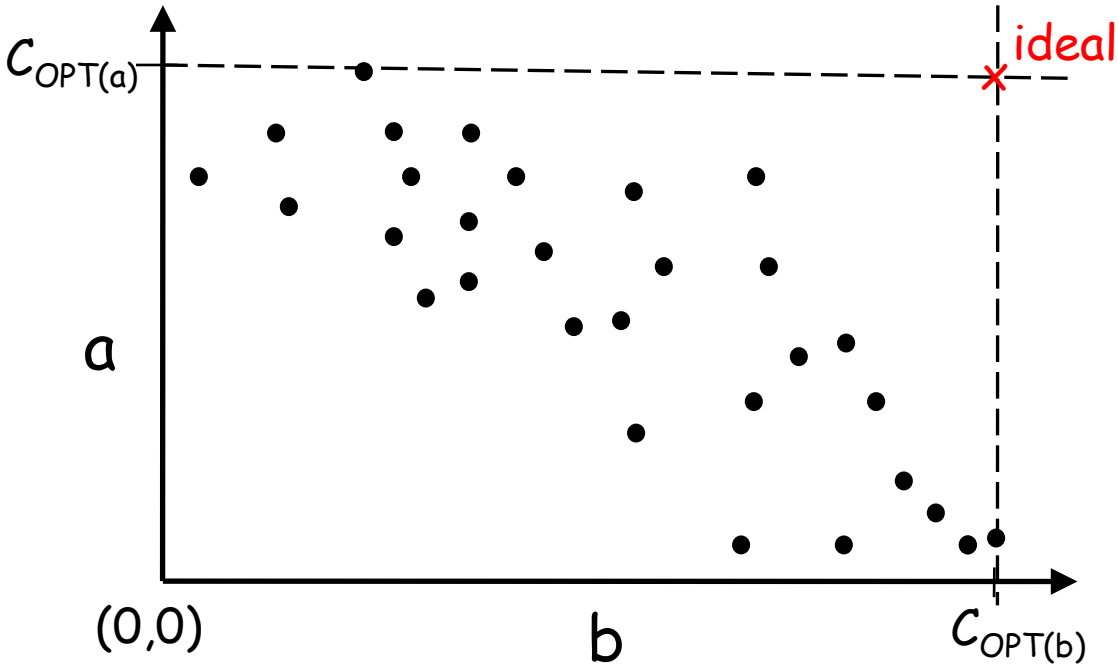
Biobjective problem :



r-approximate algo = algo r-approximate on each objective.

Approximation of the Max TSP

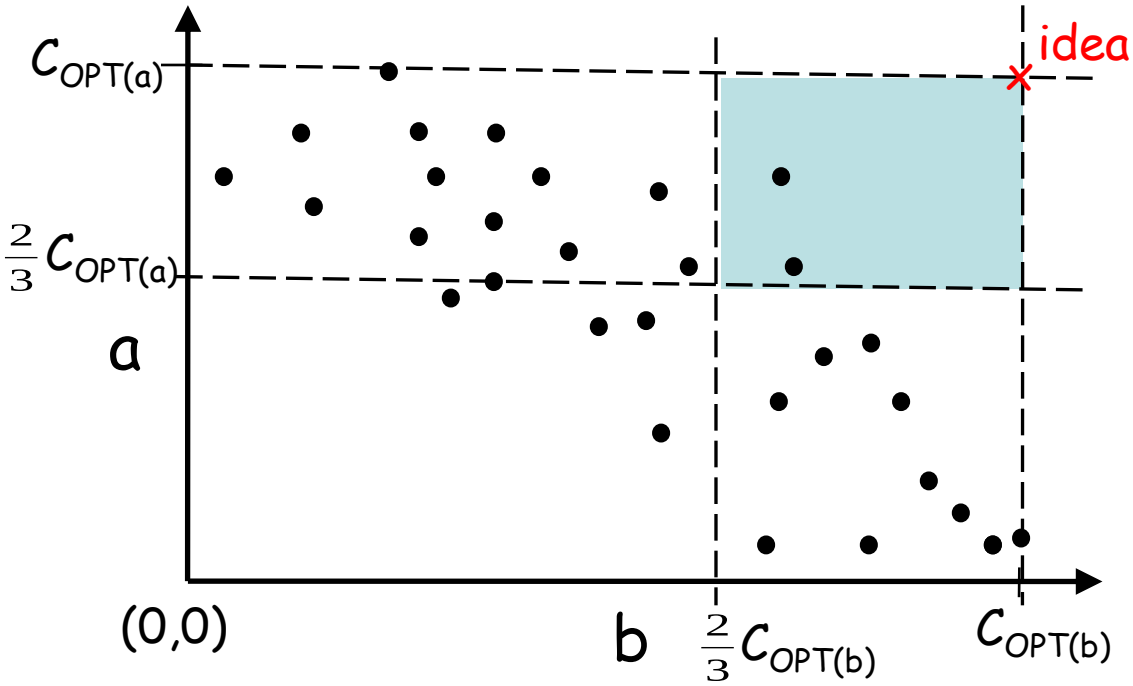
Biobjective problem :



r-approximate algo = algo r-approximate on each objective.

Approximation of the Max TSP

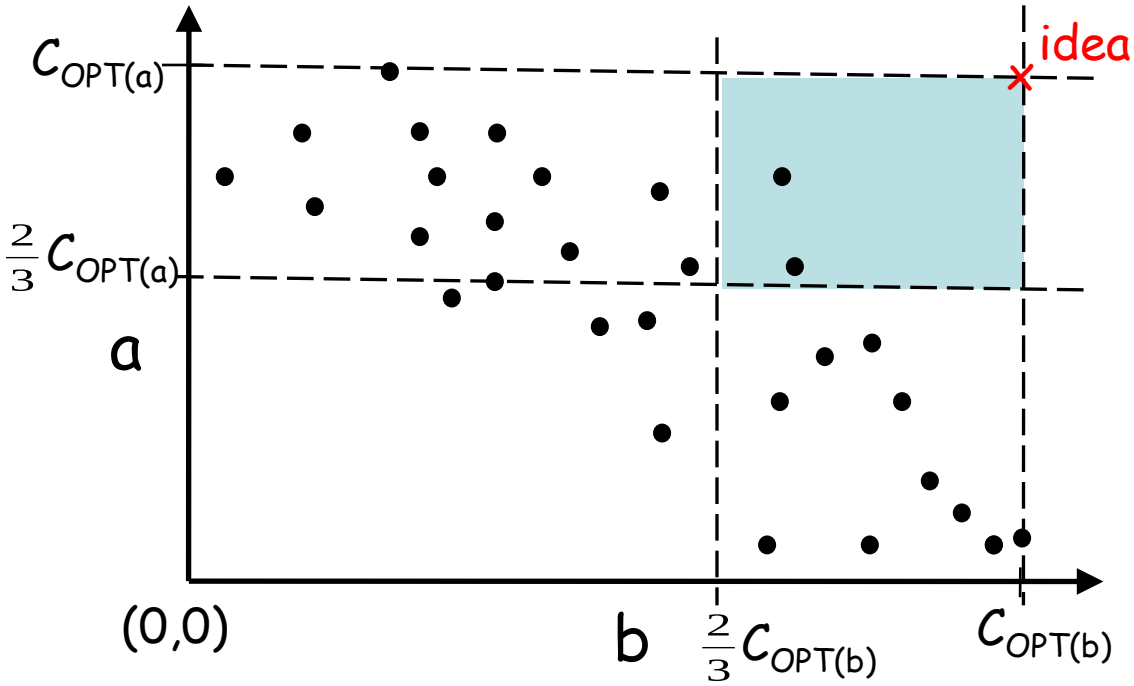
Biobjective problem :



r-approximate algo = algo r-approximate on each objective.

Approximation of the Max TSP

Biobjective problem :



r-approximate algo = algo r-approximate on each objective.

What is maximum value of r such that there exists an r-approximate algorithm for the Max TSP?

Multiobj. Max TSP: state of the art

- Efficient set approach:
  - Randomized $(1/k - \varepsilon)$-approx. algo. for k objectives [Bläser et al. ESA'2008] ; improved to $(2/3 - \varepsilon)$ [Manthey, STACS'2009]
  - Deterministic $(1/2k - \varepsilon)$-approx. algo. for k objectives [Manthey, TAMC'2011]

Multiobj. Max TSP: state of the art

- Efficient set approach:
  - Randomized $(1/k - \varepsilon)$-approx. algo. for k objectives [Bläser et al. ESA'2008] ; improved to $(2/3 - \varepsilon)$ [Manthey, STACS'2009]
  - Deterministic $(1/2k - \varepsilon)$-approx. algo. for k objectives [Manthey, TAMC'2011]
- Ideal point approach (for k=2): [Manthey, STACS'2009]
  - There is no $(1/3 + \varepsilon)$-approximate algorithm
  - If there is a r-approx algo for the Max TSP $\Rightarrow$ a r/3-approximate algo for the biobjective Max TSP (i.e. 0.25-approx algo.; 0.29-approx algo. with 2 triangular inequalities).

Outline

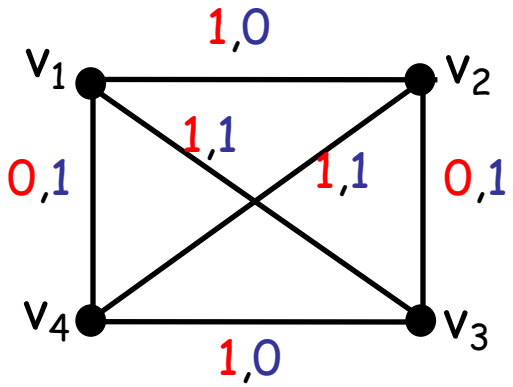
- Introduction :
  - the Multiobjective Max TSP
  - our approach
- Results for the Biobjective Max TSP
  - lower bounds
  - a generic algorithm
  - analysis of one case
- Dealing with many objectives
- Conclusion and further research

Lower bounds

- No objective fulfill the triangle inequality:
  There is no $(\frac{1}{3}+\varepsilon)$-approximate algorithm [Manthey, STACS'2009]

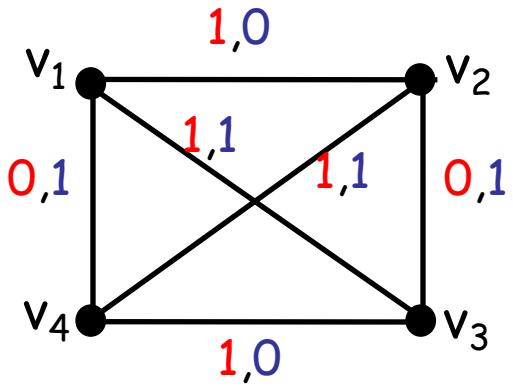
Lower bounds

- No objective fulfill the triangle inequality:
  There is no $(\frac{1}{3}+\varepsilon)$-approximate algorithm [Manthey, STACS'2009]
- Both objectives fulfills the triangle inequality:
  - No $(\frac{1}{2}+\varepsilon)$-approximate algorithm (n=4)

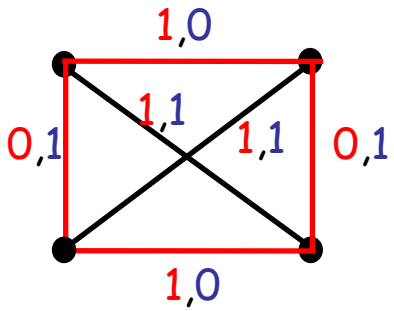


Lower bounds

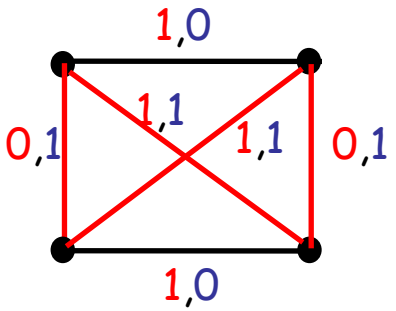
- No objective fulfill the triangle inequality:
  There is no $(\frac{1}{3}+\varepsilon)$-approximate algorithm [Manthey, STACS'2009]
- Both objectives fulfills the triangle inequality:
  - No $(\frac{1}{2}+\varepsilon)$-approximate algorithm (n=4)



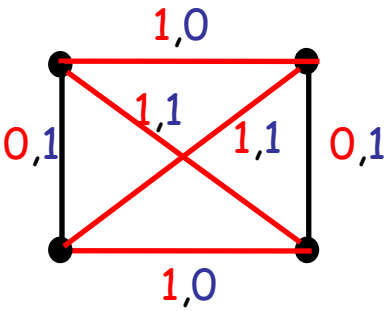
3 possible cycles :



Cost=(2,2)



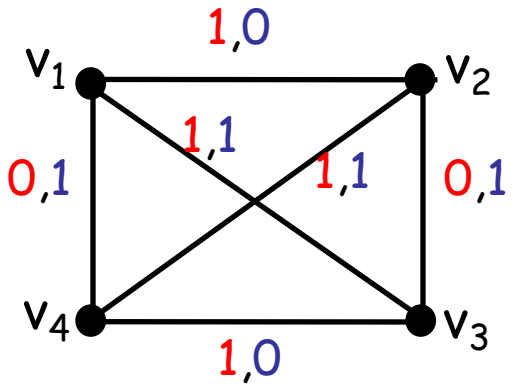
Cost=(2,4)



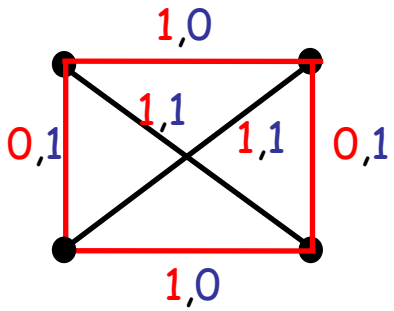
Cost=(4,2)

Lower bounds

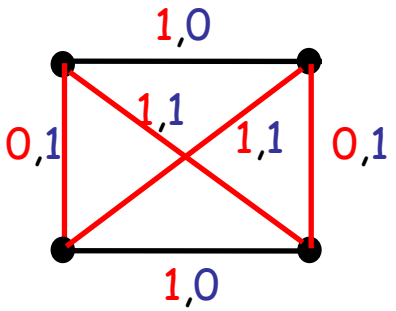
- No objective fulfill the triangle inequality:
  There is no $(\frac{1}{3}+\varepsilon)$-approximate algorithm [Manthey, STACS'2009]
- Both objectives fulfills the triangle inequality:
  - No $(\frac{1}{2}+\varepsilon)$-approximate algorithm (n=4)



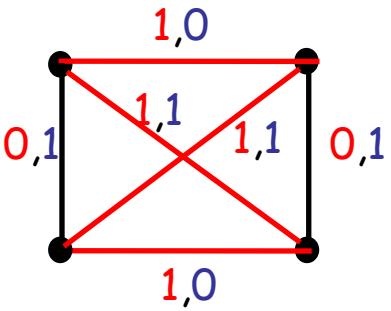
3 possible cycles :



Cost=(2,2)



Cost=(2,4)



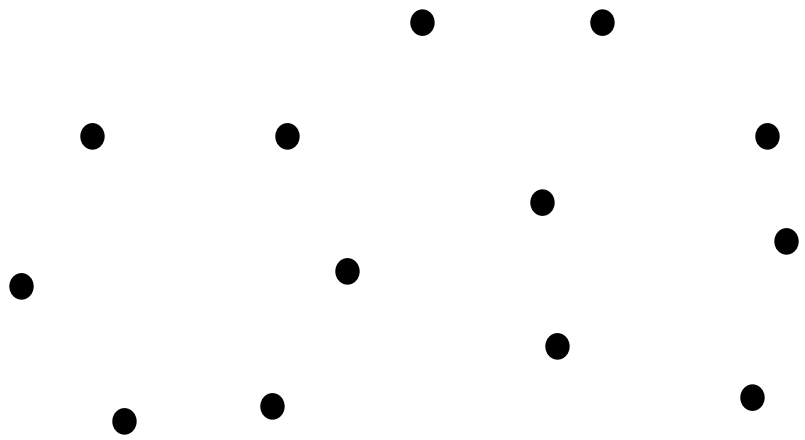
Cost=(4,2)

  - No $(\frac{3}{4}+\varepsilon)$-approximate algorithm (n large)

A generic algorithm

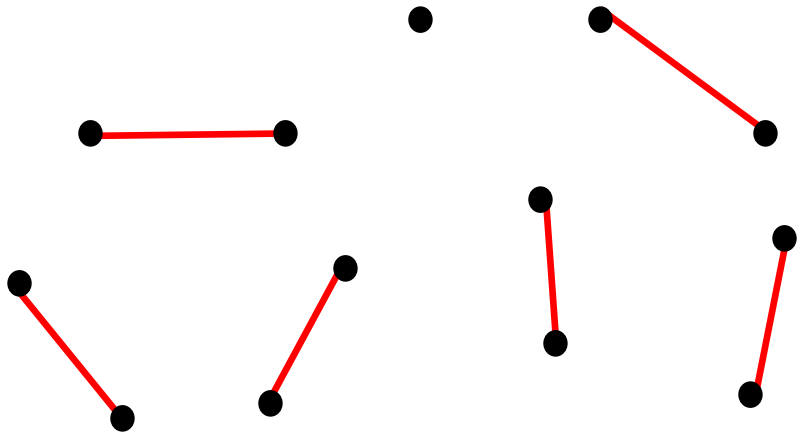
A generic algorithm

1) Build a maximum weight matching, M_a, for objective a. Do the same for objective b: M_b .



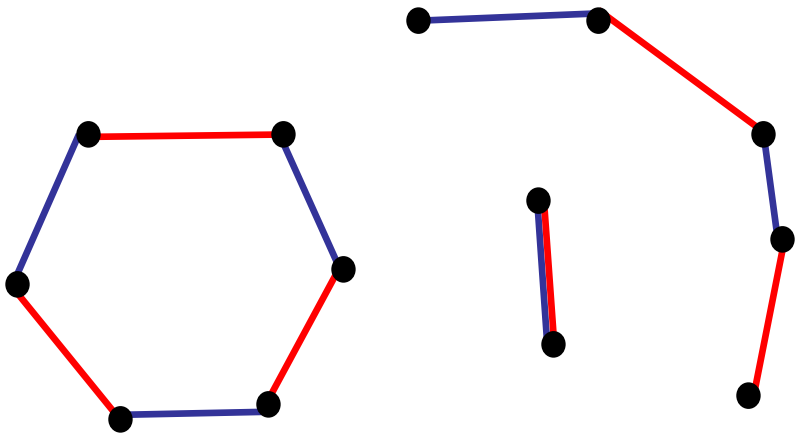
A generic algorithm

1) Build a maximum weight matching, M_a, for objective a. Do the same for objective b: M_b .



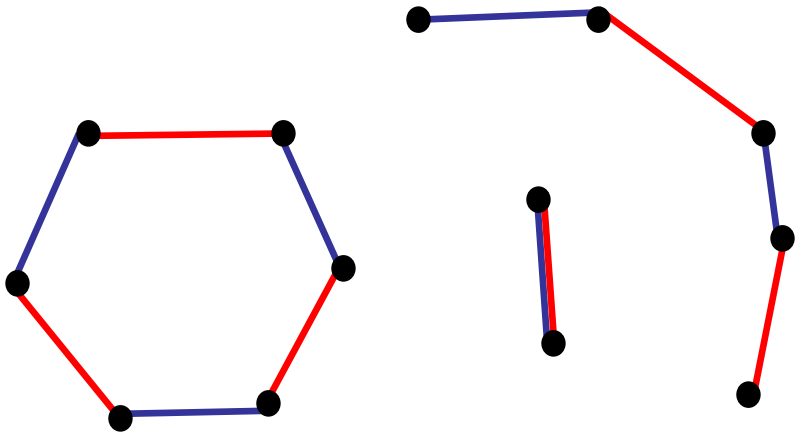
A generic algorithm

1) Build a maximum weight matching, M_a, for objective a. Do the same for objective b: M_b .



A generic algorithm

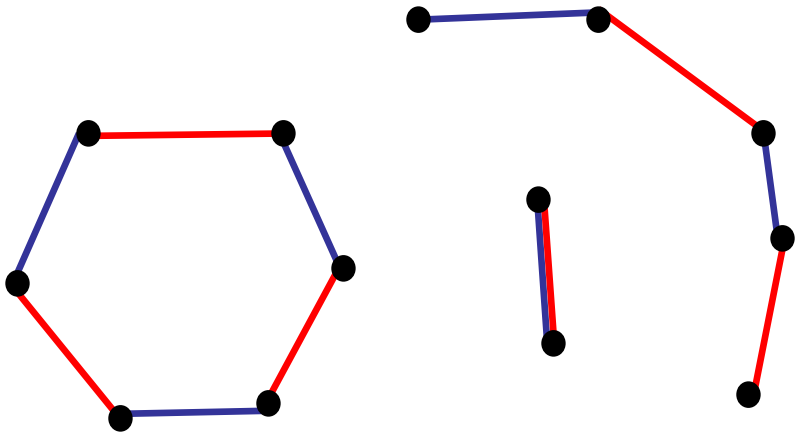
1) Build a maximum weight matching, M_a, for objective a. Do the same for objective b: M_b .



1) We may obtain : a set of cycles of even length; paths of length 1; one path of even length.

A generic algorithm

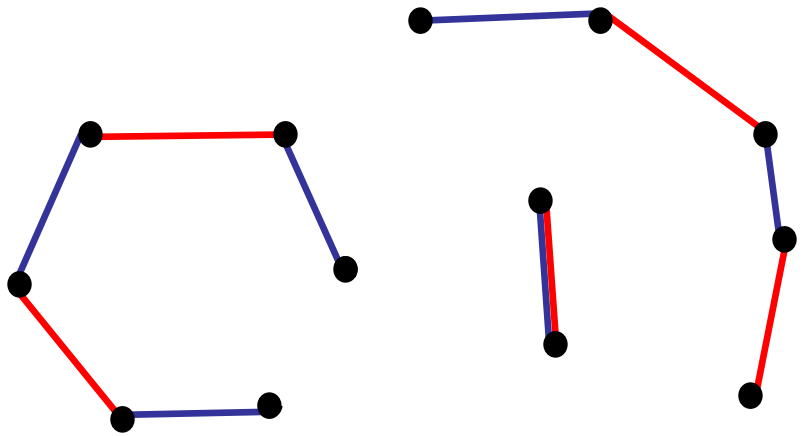
1) Build a maximum weight matching, M_a, for objective a. Do the same for objective b: M_b .
2) For each cycle C_i: remove the edge in $C_i \cap M_a$ which has the minimum weight for objective a.



1) We may obtain : a set of cycles of even length; paths of length 1; one path of even length.

A generic algorithm

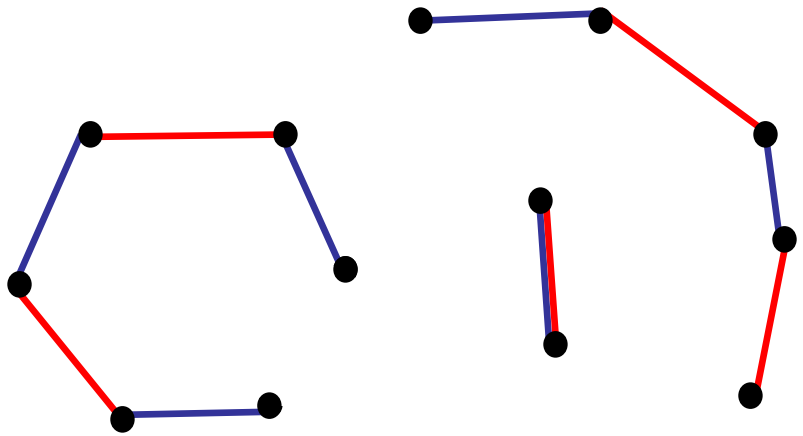
1) Build a maximum weight matching, M_a, for objective a. Do the same for objective b: M_b .
2) For each cycle C_i: remove the edge in $C_i \cap M_a$ which has the minimum weight for objective a.



1) We may obtain : a set of cycles of even length; paths of length 1; one path of even length.

A generic algorithm

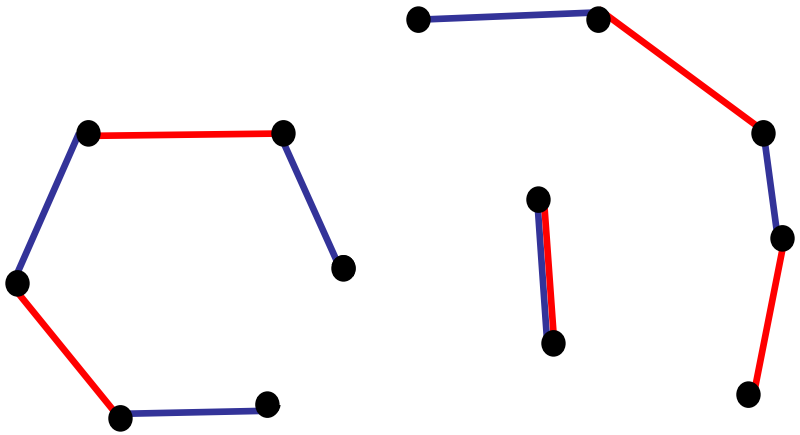
1) Build a maximum weight matching, M_a, for objective a. Do the same for objective b: M_b .
2) For each cycle C_i: remove the edge in $C_i \cap M_a$ which has the minimum weight for objective a.



1) We may obtain : a set of cycles of even length; paths of length 1; one path of even length.
2) We obtain a partial tour

A generic algorithm

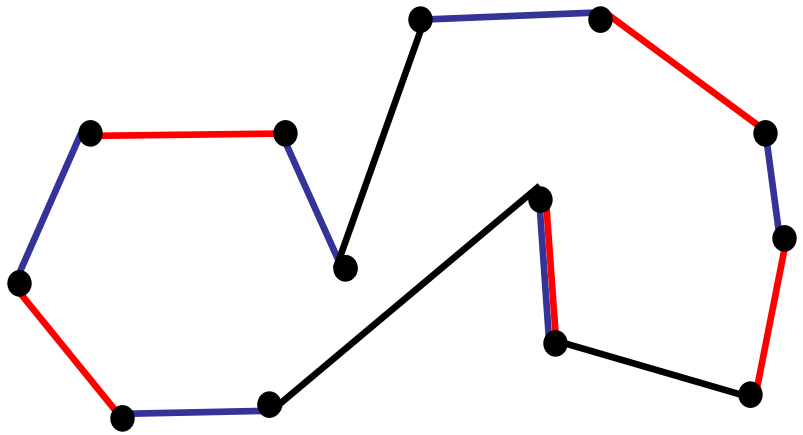
1) Build a maximum weight matching, M_a, for objective a. Do the same for objective b: M_b .
2) For each cycle C_i: remove the edge in $C_i \cap M_a$ which has the minimum weight for objective a.
3) Add edges in order to obtain an hamiltonian cycle.



1) We may obtain : a set of cycles of even length; paths of length 1; one path of even length.
2) We obtain a partial tour

A generic algorithm

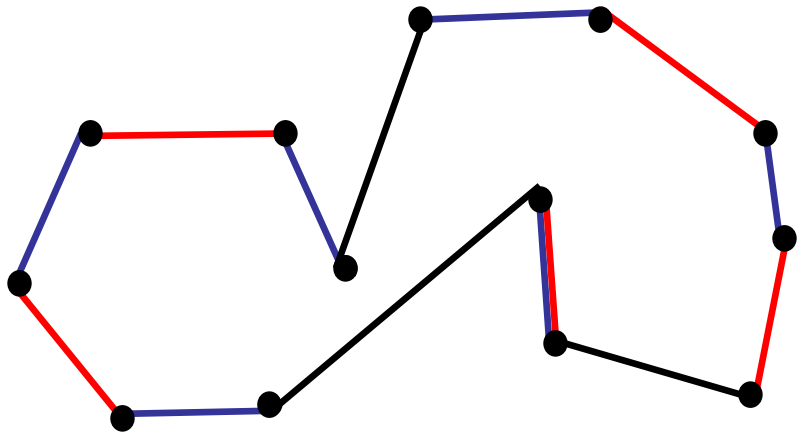
1) Build a maximum weight matching, M_a, for objective a. Do the same for objective b: M_b .
2) For each cycle C_i: remove the edge in $C_i \cap M_a$ which has the minimum weight for objective a.
3) Add edges in order to obtain an hamiltonian cycle.



1) We may obtain : a set of cycles of even length; paths of length 1; one path of even length.
2) We obtain a partial tour

A generic algorithm

1) Build a maximum weight matching, M_a, for objective a. Do the same for objective b: M_b .
2) For each cycle C_i: remove the edge in $C_i \cap M_a$ which has the minimum weight for objective a.
3) Add edges in order to obtain an hamiltonian cycle.



1) We may obtain : a set of cycles of even length; paths of length 1; one path of even length.
2) We obtain a partial tour
3) We return this cycle.

Performance of this algorithm

We show that $\begin{cases} a(\text{cycle obtained}) \geq \alpha\ a(M_a) \\ b(\text{cycle obtained}) \geq \alpha\ b(M_b) \end{cases}$

Exemple: α=1/2 in the general case.

(Proof: we remove the edge with min value on a on each cycle
$\Rightarrow$ loss $\leq a(M_a)/2$)

Performance of this algorithm

We show that $\begin{cases} a(\text{cycle obtained}) \geq \alpha\ a(M_a) \\ b(\text{cycle obtained}) \geq \alpha\ b(M_b) \end{cases}$

Exemple: α=1/2 in the general case.

(Proof: we remove the edge with min value on a on each cycle
$\Rightarrow$ loss $\leq a(M_a)/2$)

Property: If $\begin{cases} a(\text{cycle obtained}) \geq \alpha\ a(M_a) \\ b(\text{cycle obtained}) \geq \alpha\ b(M_b)\ , \end{cases}$

then the algorithm is $(\alpha/2)$-approximate.

Exemple: This algorithm is $\frac{1}{4}$-approximate.

Performance of this algorithm

We show that $\begin{cases} a(\text{cycle obtained}) \geq \alpha\ a(M_a) \\ b(\text{cycle obtained}) \geq \alpha\ b(M_b) \end{cases}$

Exemple: α=1/2 in the general case.

(Proof: we remove the edge with min value on a on each cycle
$\Rightarrow$ loss $\leq a(M_a)/2$)

Property: If $\begin{cases} a(\text{cycle obtained}) \geq \alpha\ a(M_a) \\ b(\text{cycle obtained}) \geq \alpha\ b(M_b)\ , \end{cases}$

then the algorithm is (α/2)-approximate.

Exemple: This algorithm is $\frac{1}{4}$-approximate.

Property: This algorithm is $\frac{1+2\sqrt{2}}{14} \cong 0.27$ -approximate in the general case.

With the triangular inequality on a

Property: This algorithm is 3/8-approximate.

Proof: We show that $a(\text{output}) \geq 3/4\ a(M_a)$ (same for b).

With the triangular inequality on a

Property: This algorithm is 3/8-approximate.

Proof: We show that $a(\text{output}) \geq 3/4\ a(M_a)$ (same for b).

If at the end of Step (1):

- There is no cycle: $a(\text{output}) \geq a(M_a)$

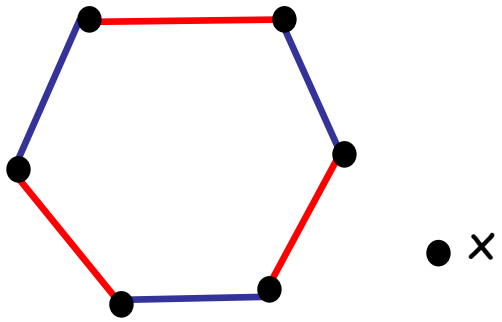
With the triangular inequality on a

Property: This algorithm is 3/8-approximate.

Proof: We show that $a(\text{output}) \geq 3/4\ a(M_a)$ (same for b).

If at the end of Step (1):

- There is no cycle: $a(\text{output}) \geq a(M_a)$
- There is one cycle and no path:
  - with n vertices: $a(\text{output}) \geq a(M_a)$
  - with (n-1) vertices:



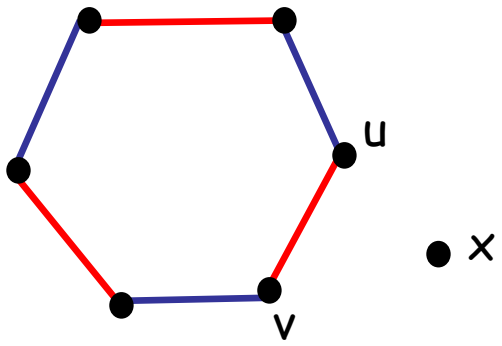
With the triangular inequality on a

Property: This algorithm is 3/8-approximate.

Proof: We show that $a(\text{output}) \geq 3/4\ a(M_a)$ (same for b).

If at the end of Step (1):

- There is no cycle: $a(\text{output}) \geq a(M_a)$
- There is one cycle and no path:
  - with n vertices: $a(\text{output}) \geq a(M_a)$
  - with (n-1) vertices:



With the triangular inequality on a

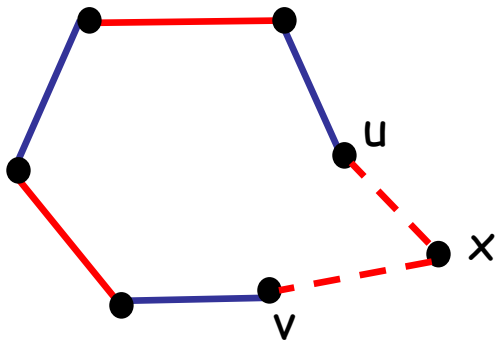
Property: This algorithm is 3/8-approximate.

Proof: We show that $a(\text{output}) \geq 3/4\ a(M_a)$ (same for b).

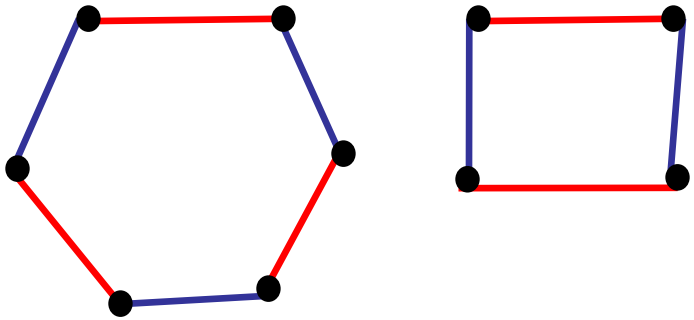
If at the end of Step (1):

- There is no cycle: $a(\text{output}) \geq a(M_a)$
- There is one cycle and no path:
  - with n vertices: $a(\text{output}) \geq a(M_a)$
  - with (n-1) vertices:

$a(\text{output}) \geq a(M_a)$

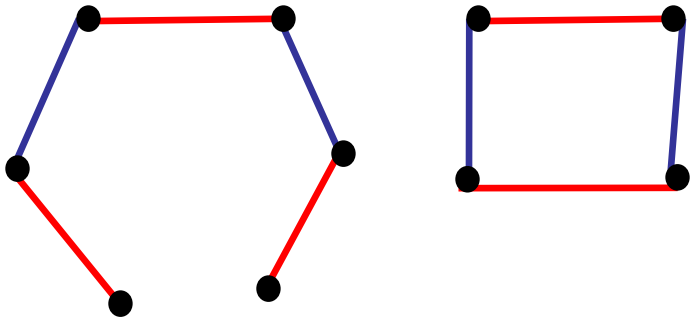


- There are several cycles and no path:



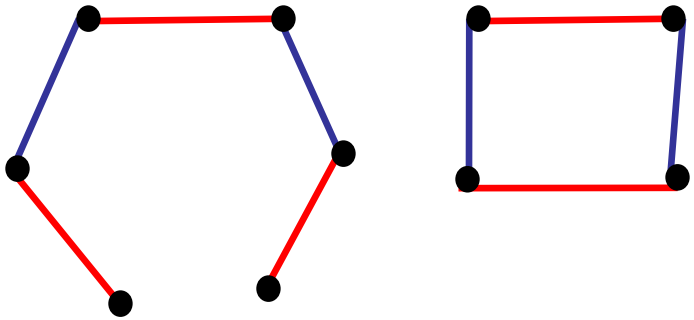
- Remove the edge with smallest weight on b.

- There are several cycles and no path:



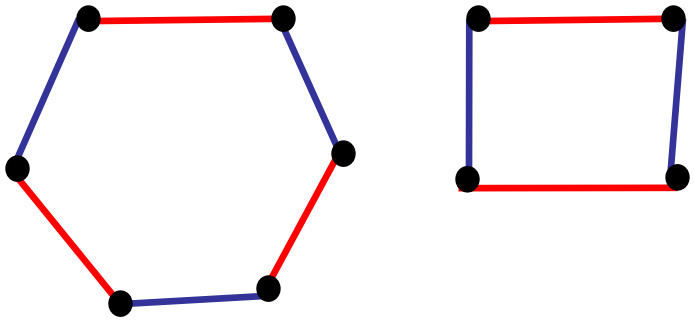
- Remove the edge with smallest weight on b.

- There are several cycles and no path:



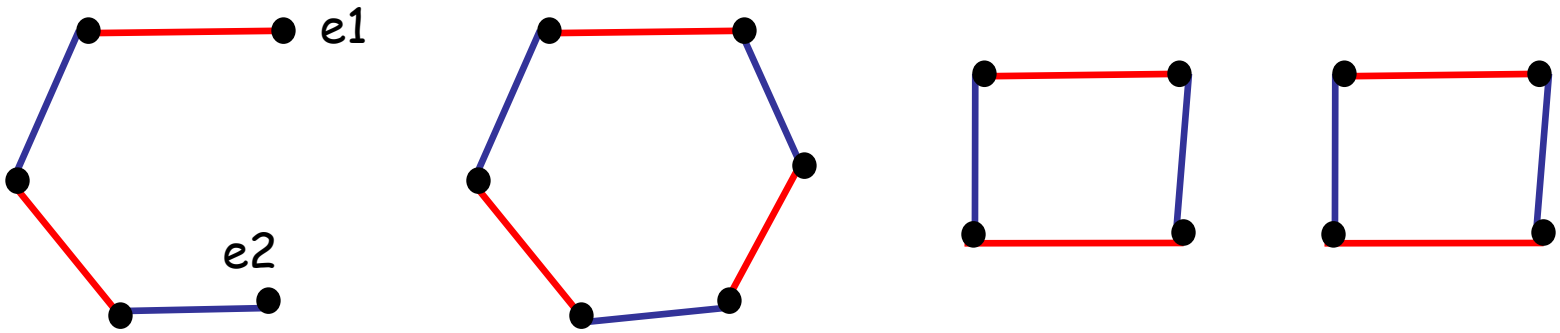
- Remove the edge with smallest weight on b.
$\Rightarrow$ the cost of the output is $\geq 3/4\ b(M_b)$.
- Go to the next case.

- There are several cycles and no path:

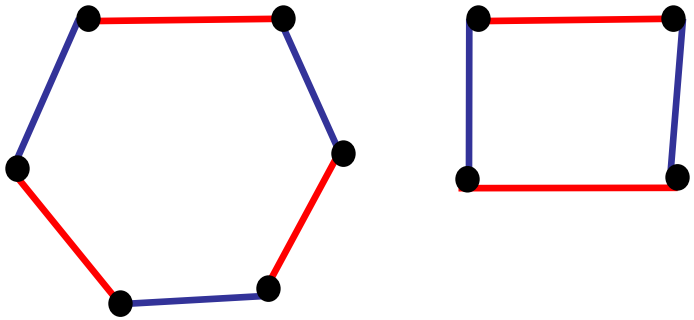


- Remove the edge with smallest weight on b.
$\Rightarrow$ the cost of the output is $\geq 3/4\ b(M_b)$.
- Go to the next case.

- There is at least one cycle and one path:

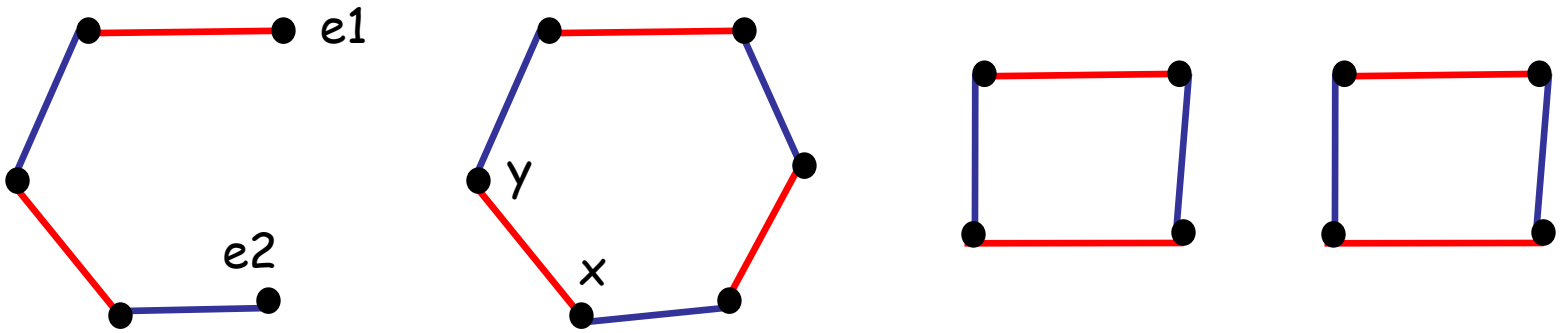


- There are several cycles and no path:



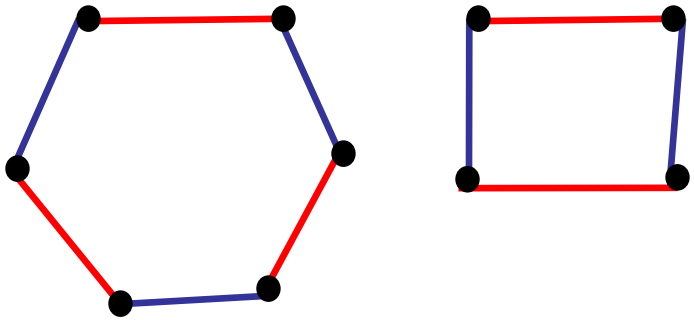
- Remove the edge with smallest weight on b.
$\Rightarrow$ the cost of the output is $\geq 3/4\ b(M_b)$.
- Go to the next case.

- There is at least one cycle and one path:



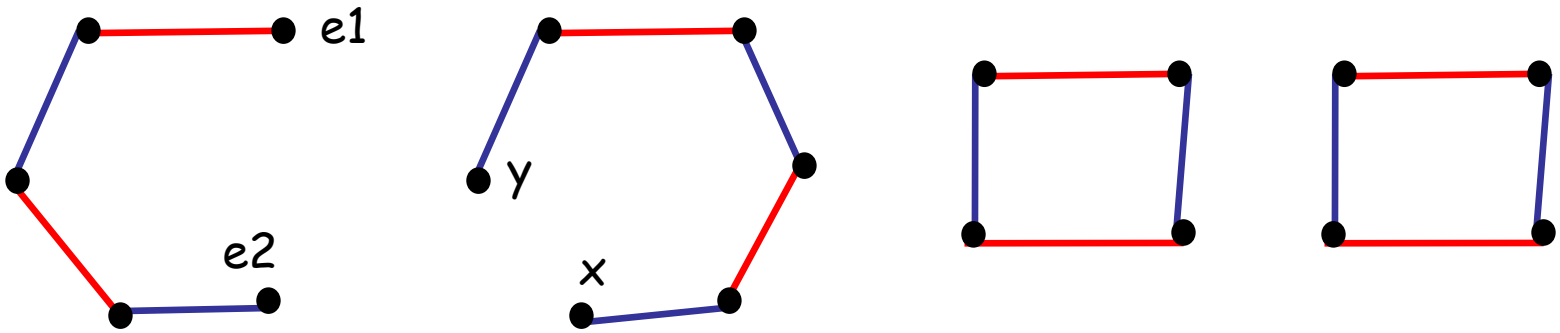
For cycle C_1 : - delete the edge with smallest value on a.

- There are several cycles and no path:



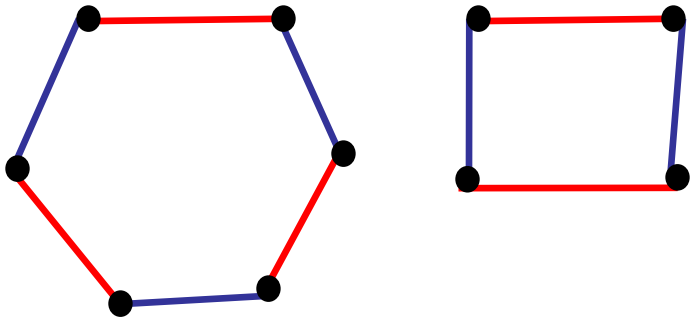
- Remove the edge with smallest weight on b.
$\Rightarrow$ the cost of the output is $\geq 3/4\ b(M_b)$.
- Go to the next case.

- There is at least one cycle and one path:



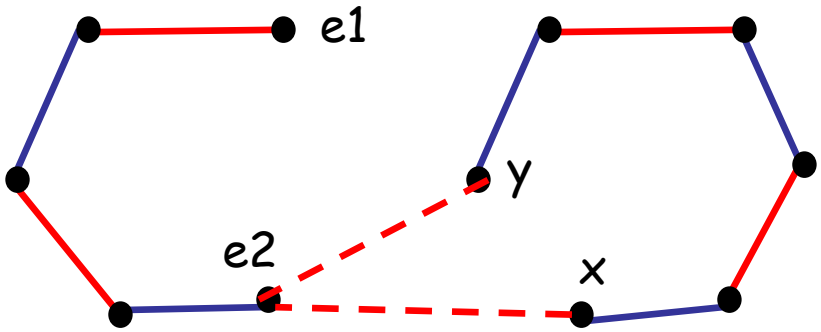
For cycle C_1 : - delete the edge with smallest value on a.

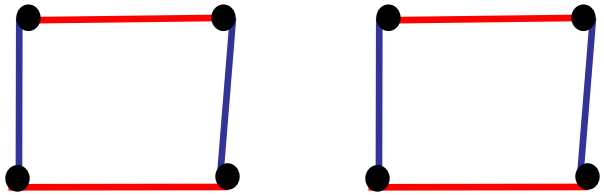
- There are several cycles and no path:



- Remove the edge with smallest weight on b.
$\Rightarrow$ the cost of the output is $\geq 3/4\ b(M_b)$.
- Go to the next case.

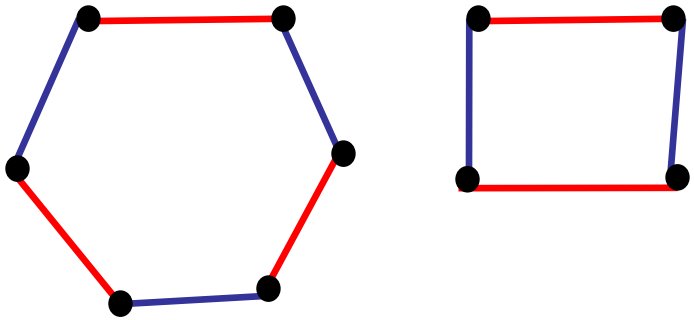
- There is at least one cycle and one path:





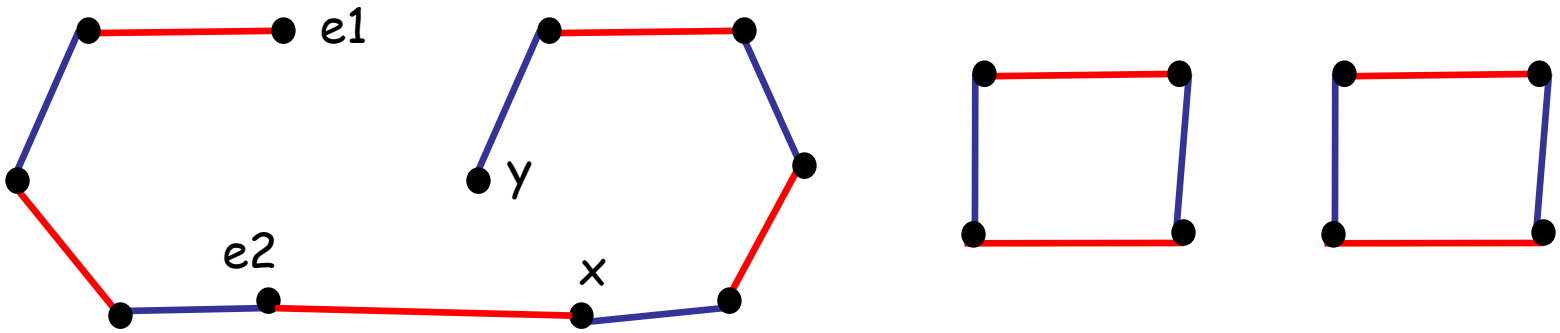
For cycle C_1 : - delete the edge with smallest value on a.
- replace it by {x,e2} or {y,e2} (take the one which has the best value on a)

- There are several cycles and no path:



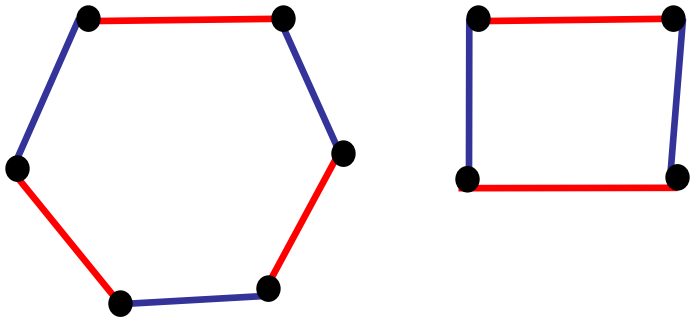
- Remove the edge with smallest weight on b.
$\Rightarrow$ the cost of the output is $\geq 3/4\ b(M_b)$.
- Go to the next case.

- There is at least one cycle and one path:



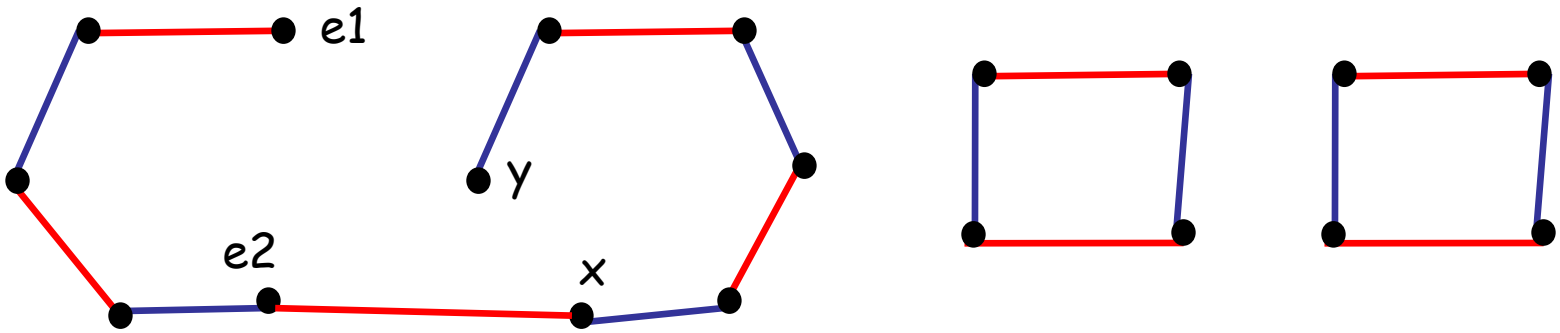
For cycle C_1 : - delete the edge with smallest value on a.
- replace it by {x,e2} or {y,e2} (take the one which has the best value on a)

- There are several cycles and no path:



- Remove the edge with smallest weight on b.
$\Rightarrow$ the cost of the output is $\geq 3/4\ b(M_b)$.
- Go to the next case.

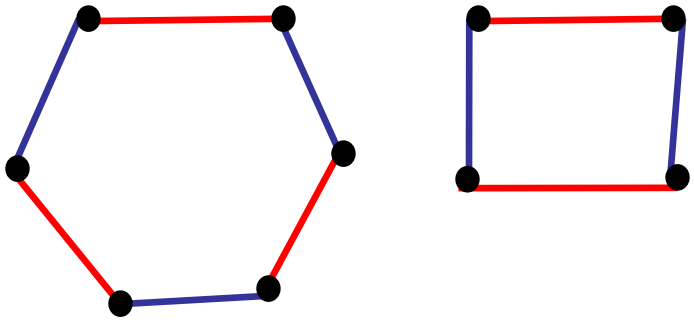
- There is at least one cycle and one path:



For cycle C_1 : - delete the edge with smallest value on a.
- replace it by {x,e2} or {y,e2} (take the one which has the best value on a)

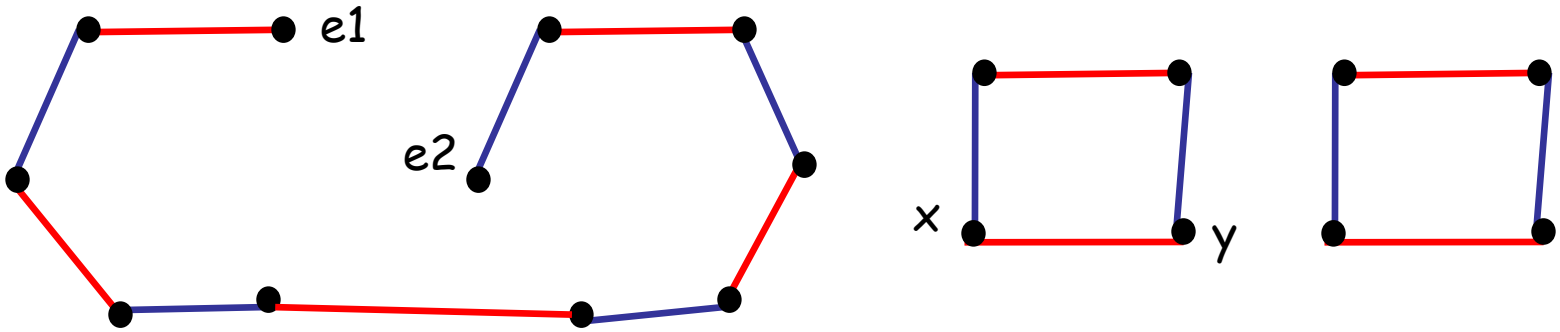
Repeat this process until there is no more cycle.

- There are several cycles and no path:



- Remove the edge with smallest weight on b.
$\Rightarrow$ the cost of the output is $\geq 3/4\ b(M_b)$.
- Go to the next case.

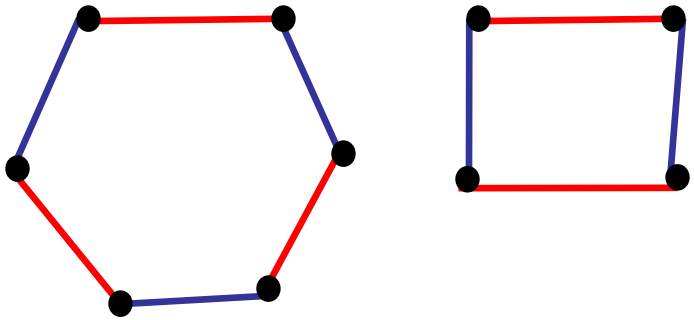
- There is at least one cycle and one path:



For cycle C_1 : - delete the edge with smallest value on a.
- replace it by {x,e2} or {y,e2} (take the one which has the best value on a)

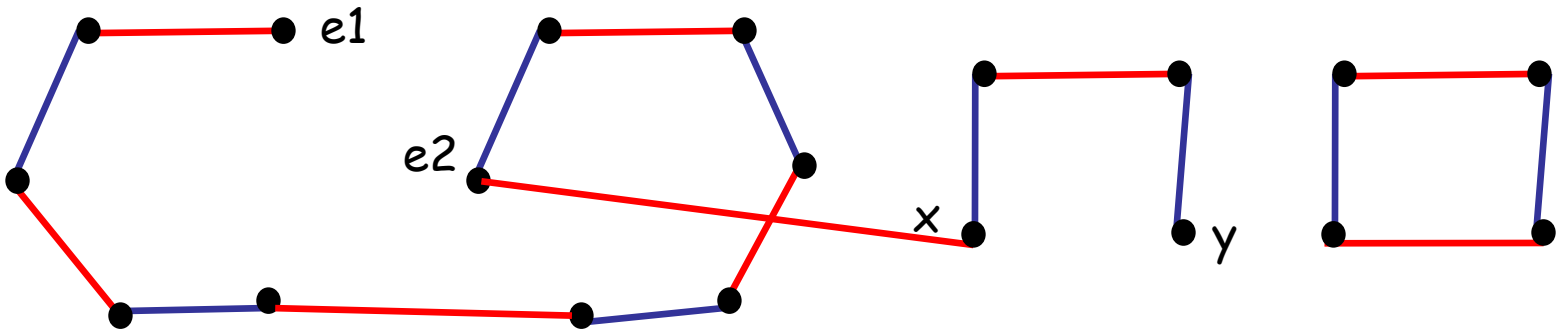
Repeat this process until there is no more cycle.

- There are several cycles and no path:



- Remove the edge with smallest weight on b.
$\Rightarrow$ the cost of the output is $\geq 3/4\ b(M_b)$.
- Go to the next case.

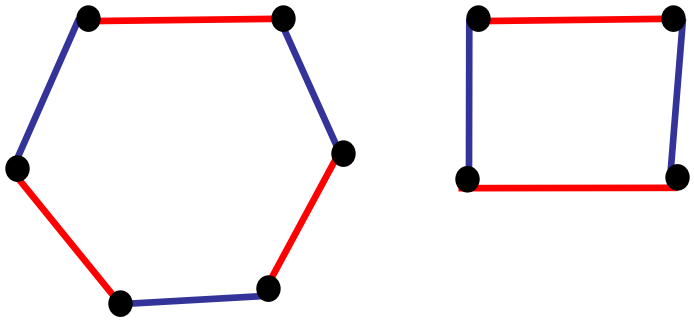
- There is at least one cycle and one path:



For cycle C_1 : - delete the edge with smallest value on a.
- replace it by {x,e2} or {y,e2} (take the one which has the best value on a)

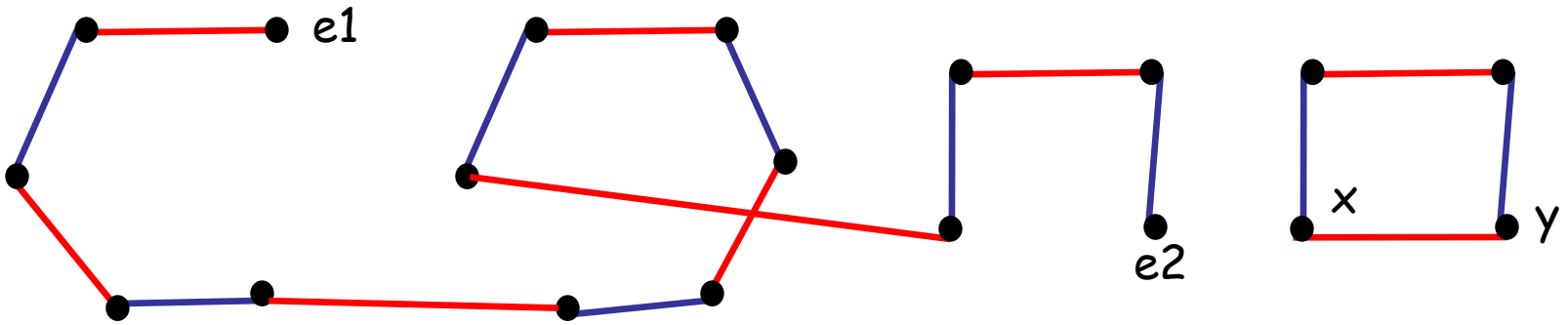
Repeat this process until there is no more cycle.

- There are several cycles and no path:



- Remove the edge with smallest weight on b.
$\Rightarrow$ the cost of the output is $\geq 3/4\ b(M_b)$.
- Go to the next case.

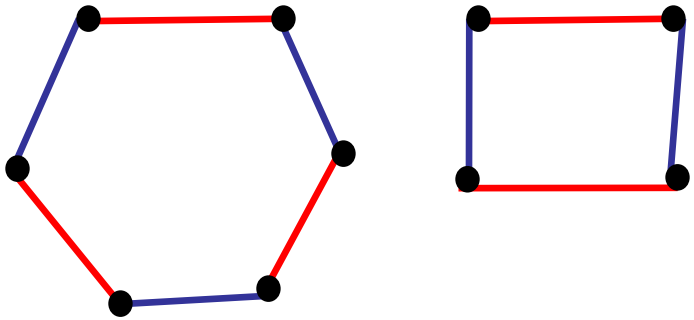
- There is at least one cycle and one path:



For cycle C_1 : - delete the edge with smallest value on a.
- replace it by {x,e2} or {y,e2} (take the one which has the best value on a)

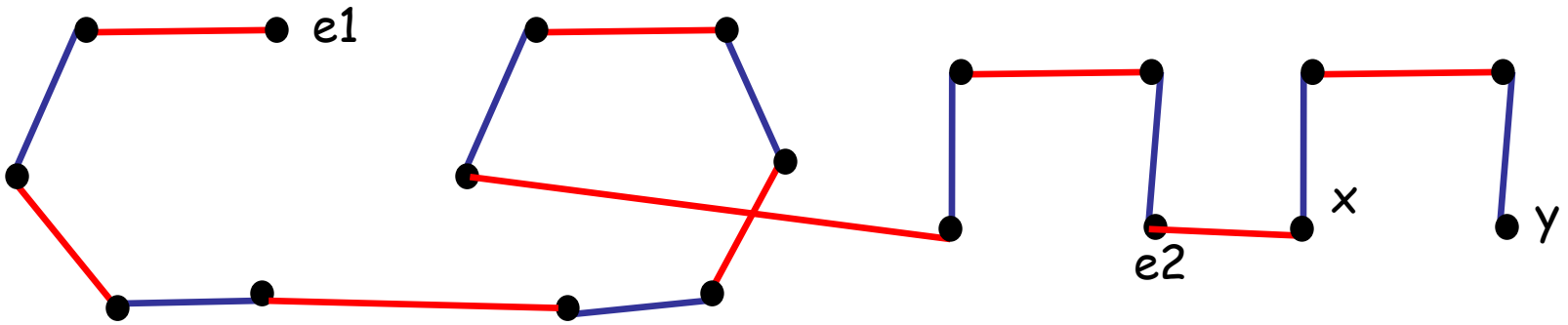
Repeat this process until there is no more cycle.

- There are several cycles and no path:



- Remove the edge with smallest weight on b.
$\Rightarrow$ the cost of the output is $\geq 3/4\ b(M_b)$.
- Go to the next case.

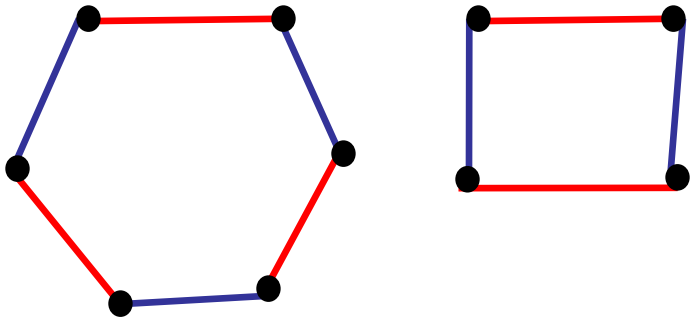
- There is at least one cycle and one path:



For cycle C_1 : - delete the edge with smallest value on a.
- replace it by {x,e2} or {y,e2} (take the one which has the best value on a)

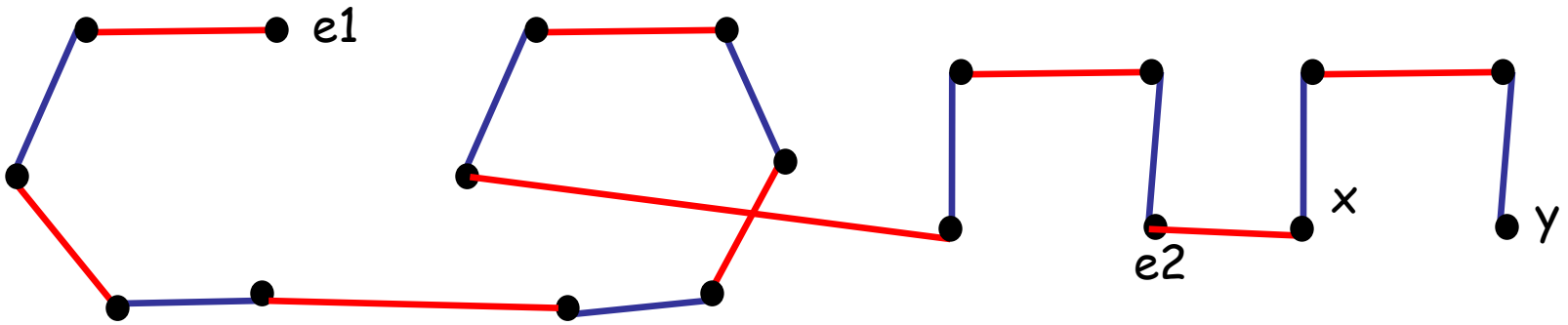
Repeat this process until there is no more cycle.

- There are several cycles and no path:



- Remove the edge with smallest weight on b.
$\Rightarrow$ the cost of the output is $\geq 3/4\ b(M_b)$.
- Go to the next case.

- There is at least one cycle and one path:



For cycle C_1 : - delete the edge with smallest value on a.
- replace it by {x,e2} or {y,e2} (take the one which has the best value on a)

Repeat this process until there is no more cycle.

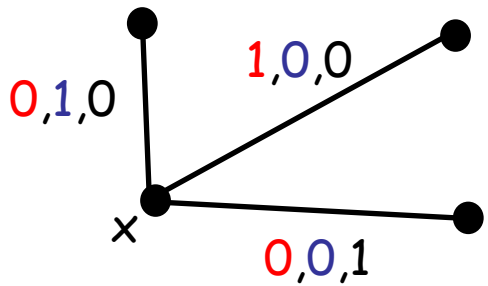
$\Rightarrow$ The cost of the output is $\geq 3/4\ a(M_a)$ and $\geq 3/4\ b(M_b)$

Outline

- Introduction :
  - the Multiobjective Max TSP
  - our approach
- Results for the Biobjective Max TSP
  - lower bounds
  - a generic algorithm
  - analysis of one case
- Dealing with many objectives
- Conclusion and further research

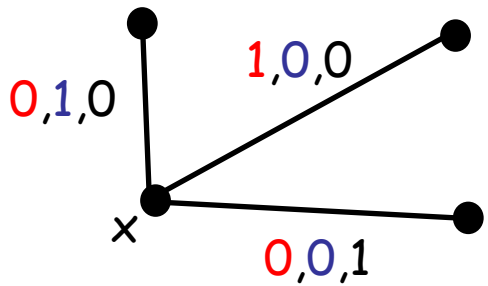
Dealing with many objectives

If no objective fulfills the triangle inequality, there is no approximate algorithm:

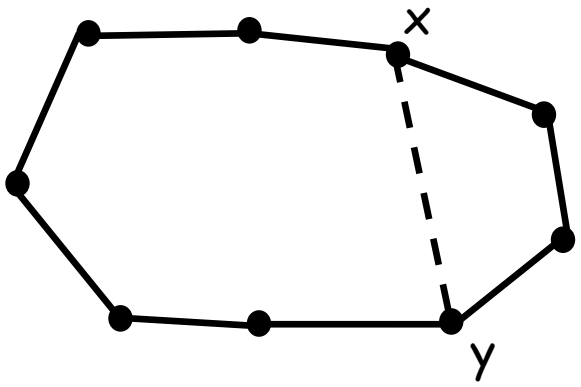


Dealing with many objectives

If no objective fulfills the triangle inequality, there is no approximate algorithm:



Property: If all the objectives fulfill the triangle inequality, then each cycle is n/2-approximate, and there is no (n/2+ε)-approximate algorithm.



Let T be any cycle.
Let T* be the optimal cycle for objective i,
and let {x,y} be its best edge : $i(x,y) \geq i(T^*)/n$.
Using the triangular inequality: $i(T) \geq 2\, i(T^*)/n$.

Conclusion and further research

- Approximation of the ideal point for the BiObjective Max TSP in 3 cases:
  - General case: $0.27 \leq r \leq 1/3$
  - One triangular inequality: $3/8 \leq r \leq 3/4$
  - Two triangular inequalities: $5/12 \leq r \leq 3/4$
- Many objectives : best approximation is n/2 (with triangular inequality)

- Intermediate cases: k-objective Max TSP?
- Multiobjective asymmetric Max TSP?
- Experimentations