Multi-criteria shortest paths

Antonin Lentz

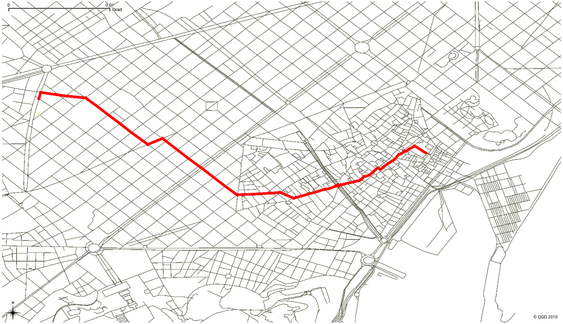
with Nicolas Hanusse and David Ilcinkas

LaBRI, France

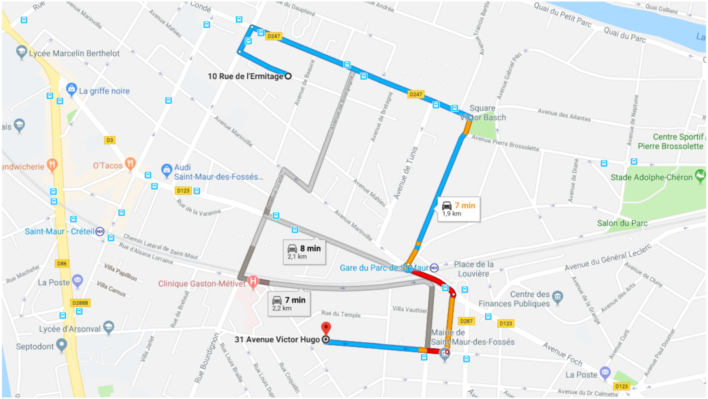
Query

Compute the "best" path

- from a source vertex s
- to a target vertex t

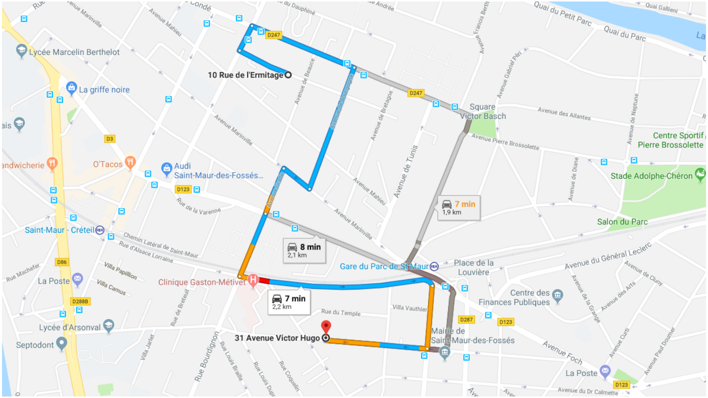


Shortest path problem on a road network



Shortest path computation with Google Maps

Introduction



Shortest path computation with Google Maps

Classical methods

Let G be a weighted graph with n vertices and m edges.

Direct methods

- Bellman Ford: $\mathcal{O}(m \times n)$ (1956)
- Dijkstra: $\mathcal{O}(m \times \log n)$ (1959)

Classical methods

Let G be a weighted graph with n vertices and m edges.

Direct methods

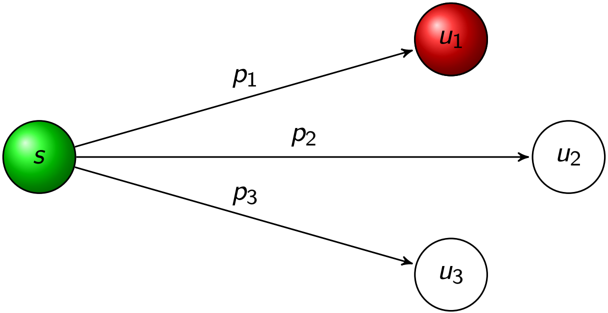
$$\left.\begin{array}{l}\bullet\ \text{Bellman Ford: } \mathcal{O}(m \times n)\ (1956) \\ \bullet\ \text{Dijkstra: } \mathcal{O}(m \times \log n)\ (1959)\end{array}\right\}\text{one-to-all algorithms}$$

Classical methods

Let G be a weighted graph with n vertices and m edges.

Direct methods

$$\left.\begin{array}{l} \bullet \text{ Bellman Ford: } \mathcal{O}(m \times n) \text{ (1956)} \\ \bullet \text{ Dijkstra: } \mathcal{O}(m \times \log n) \text{ (1959)} \end{array}\right\} \text{one-to-all algorithms}$$

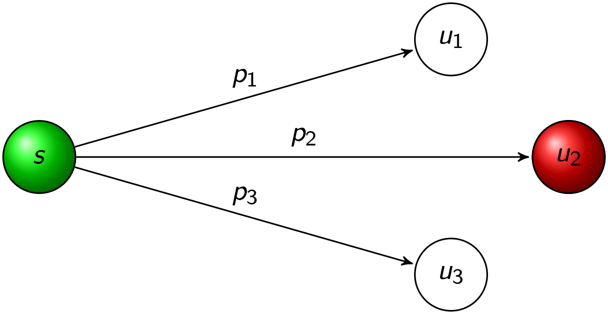


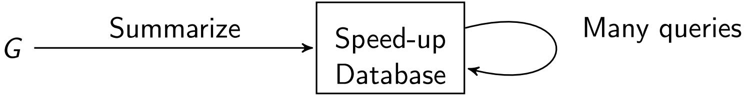
Classical methods

Let G be a weighted graph with n vertices and m edges.

Direct methods

$$\left.\begin{array}{l}\bullet\ \text{Bellman Ford: } \mathcal{O}(m \times n)\ (1956)\\ \bullet\ \text{Dijkstra: } \mathcal{O}(m \times \log n)\ (1959)\end{array}\right\}\text{one-to-all algorithms}$$



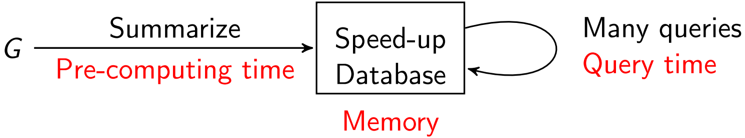


Speed-up utilization

Speed-up ideas and algorithms

- landmarks: TNR,
- separators: CRP, HPML,
- hierarchical techniques: CH, CCH, Reach
- and many others.

Speed-up techniques

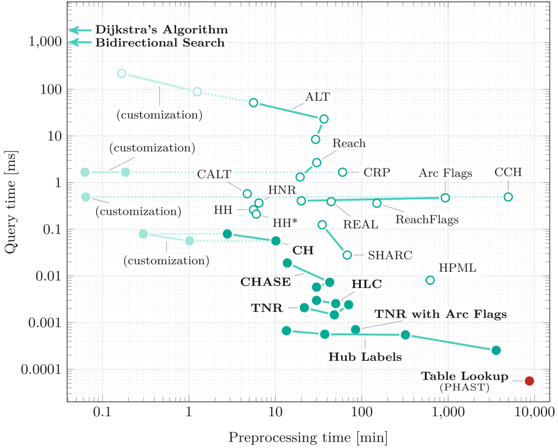


Speed-up utilization

Speed-up ideas and algorithms

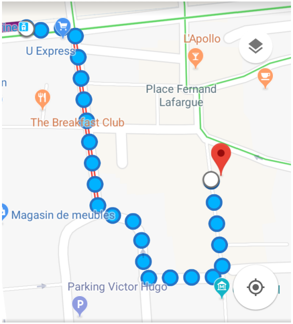
- landmarks: TNR,
- separators: CRP, HPML,
- hierarchical techniques: CH, CCH, Reach
- and many others.

Speed-up techniques



Comparison of different methods [Bast et al., 2016]

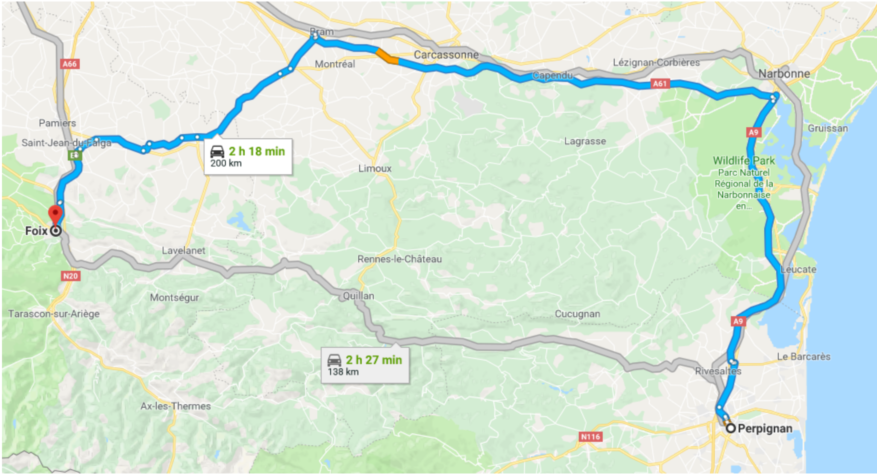
Western Europe: 18.0 millions vertices and 42.5 millions edges



Shortest path problem on a road network

Query time : at least few seconds for a 7.10^6 nodes graph [Wagner and Zündorf, 2017]

Multi-criteria Unimodal



Shortest path problem on a road network

Multi-criteria Multimodal

Multimodal network may imply multi-criteria

Taking into account other transportation means naturally introduces a lot of criteria:

- time,
- distance,
- price,
- number of transitions [Delling et al., 2014],
- difficulty (bike) [Hrnčíř et al., 2017],
- uncertainty,
- ...

Modelisation

For d criteria, weights are d dimensional vectors $(w_1, ... w_d)$, $\mathbf{w_i} \geq \mathbf{1}$.

What is the "best"?

Goal : compare two vectors of weights $(x_i)_i$ and $(y_i)_i$

What is the "best"?

Goal : compare two vectors of weights $(x_i)_i$ and $(y_i)_i$

Linear combinaison

→ total order, same as before :

TRANSIT Algorithm [Antsfeld and Walsh, 2012]

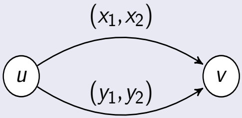
What is the "best"?

Goal : compare two vectors of weights $(x_i)_i$ and $(y_i)_i$

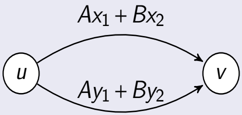
Linear combinaison

→ total order, same as before :

TRANSIT Algorithm [Antsfeld and Walsh, 2012]



Two dimensionnal edges



Linear combinaison edges

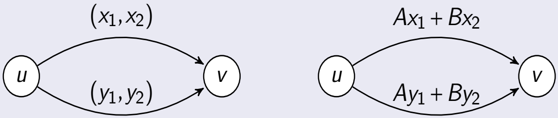
What is the "best"?

Goal : compare two vectors of weights $(x_i)_i$ and $(y_i)_i$

Linear combinaison

→ total order, same as before :

TRANSIT Algorithm [Antsfeld and Walsh, 2012]



Two dimensionnal edges   Linear combinaison edges

⚠How to choose A, B?

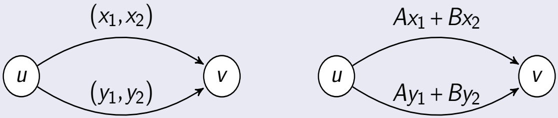
What is the "best"?

Goal : compare two vectors of weights $(x_i)_i$ and $(y_i)_i$

Linear combinaison

→ total order, same as before :

TRANSIT Algorithm [Antsfeld and Walsh, 2012]

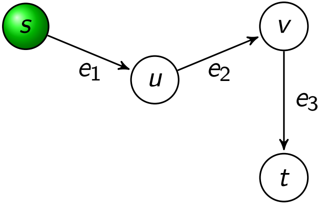


Two dimensionnal edges   Linear combinaison edges

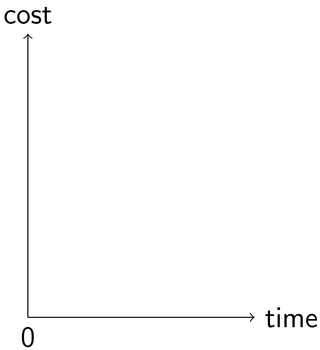
⚠ How to choose A, B?

→ Domination relation (partial order)

Domination

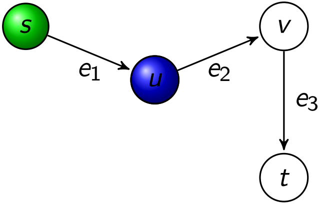


Path in a graph

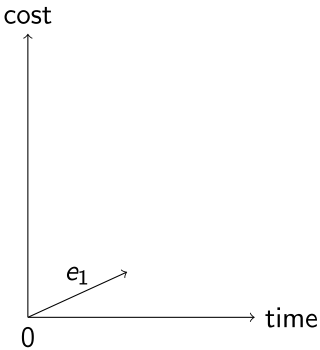


Path coordinates (2D)

Domination

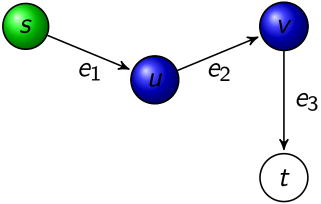


Path in a graph

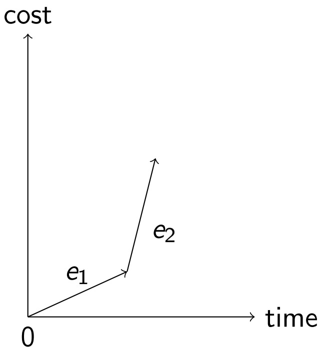


Path coordinates (2D)

Domination

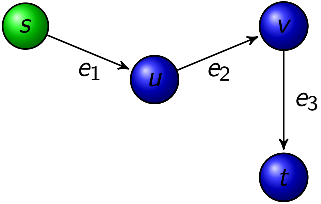


Path in a graph

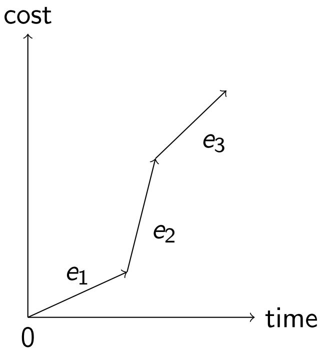


Path coordinates (2D)

Domination



Path in a graph

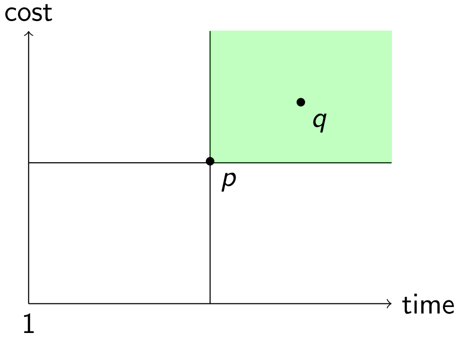


Path coordinates (2D)

Domination

$$\mathbf{p} = (p_1, ...p_d) \text{ is dominated by } \mathbf{q} = (q_1, ..., q_d) \text{ if:}$$

$$\forall i, q_i \leq p_i$$

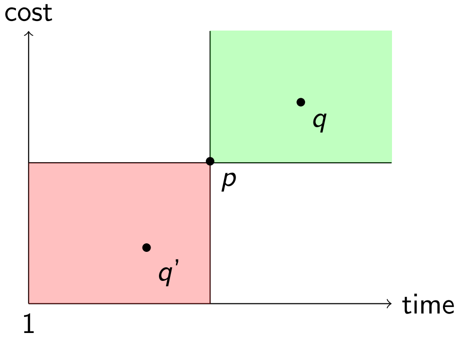


Domination area of p

Domination

$\mathbf{p} = (p_1, ...p_d)$ is dominated by $\mathbf{q} = (q_1, ..., q_d)$ if:

$$\forall i, q_i \leq p_i$$

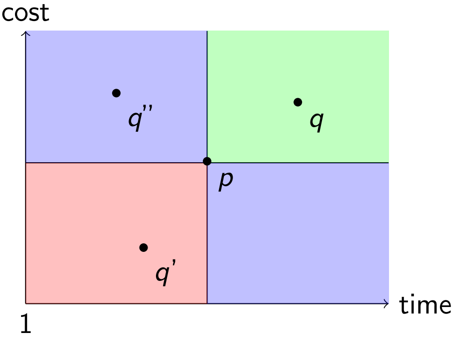


Domination area of p

Domination

$\mathbf{p} = (p_1, ...p_d)$ is dominated by $\mathbf{q} = (q_1, ..., q_d)$ if:

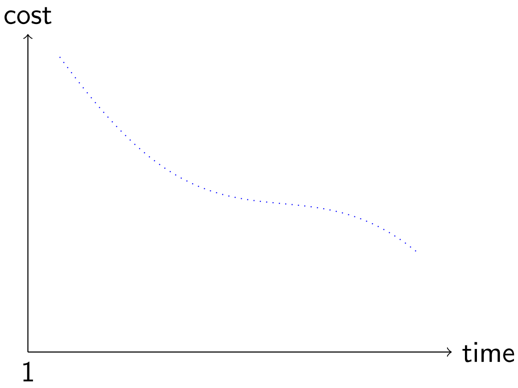
$$\forall i, q_i \leq p_i$$



Domination area of p

Pareto Set

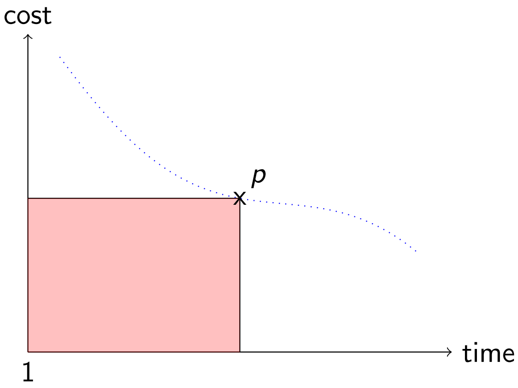
Goal: obtain non dominated solutions (Pareto Set)



Pareto set with two dimensions

Pareto Set

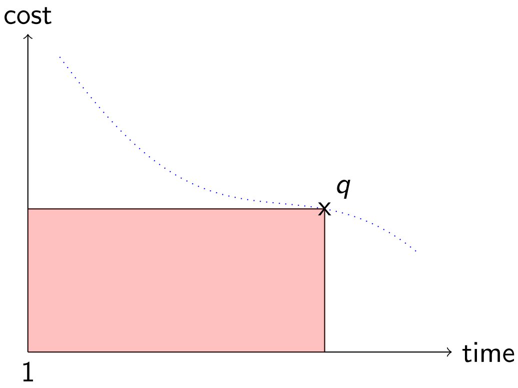
Goal: obtain non dominated solutions (Pareto Set)



Pareto set with two dimensions

Pareto Set

Goal: obtain non dominated solutions (Pareto Set)



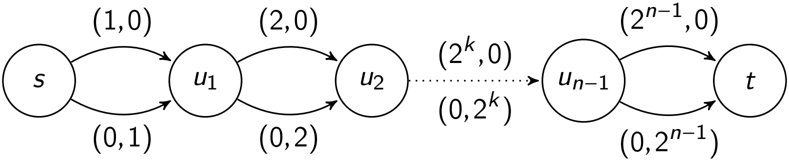
Pareto set with two dimensions

Exponential Pareto Set

Problem: too many solutions ! If :

- Δ is the maximum degree,
- n the number of vertices,

Then possibly Δ^n non-dominated solutions.



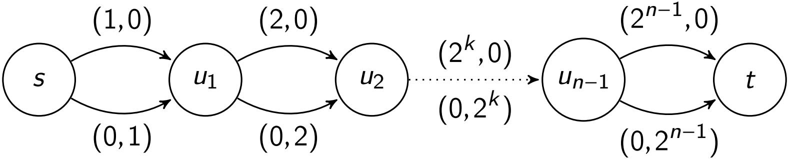
Pathological example

Exponential Pareto Set

Problem: too many solutions ! If :

- Δ is the maximum degree,
- n the number of vertices,

Then possibly Δ^n non-dominated solutions.

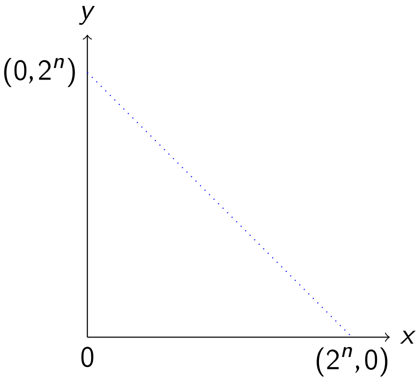


Pathological example

Every path is non dominated !

Exponential Pareto Set

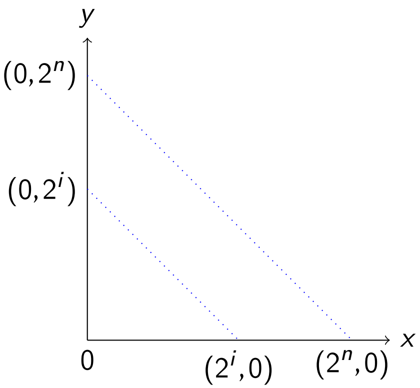
The Pareto Set is $\{(k, 2^n - 1 - k), k \in [\![0, 2^n - 1]\!]\}$



Pareto set of the vertex t

Exponential Pareto Set

The Pareto Set is $\{(k, 2^n - 1 - k), k \in [\![0, 2^n - 1]\!]\}$



Pareto set of the vertex u_i

A set back in practice too

Difficulties

- even with real datasets, algorithms are way slower !
  Examples:
  uni-criterion: micro-seconds for 10^6 node graphs [Bast et al. , 2016]
  tri-criteria: 15min for 7.10^4 node graphs [Hrnčíř et al., 2017]
- the user does not want to get a lot of propositions :
  $|ParetoSet| = 1600$ in the above example.

Summarize Pareto Sets

How to summarize?

- K-paths,
- heuristics,
- approximation,

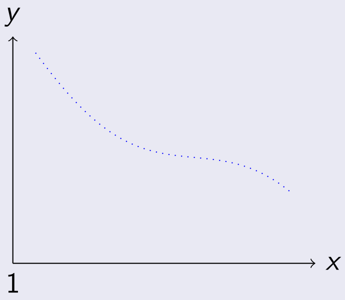
Summarize Pareto Sets

How to summarize?

- **K-paths: output only K paths from the Pareto Set**
- heuristics,
- approximation,

How to summarize?

- **K-paths: output only K paths from the Pareto Set**



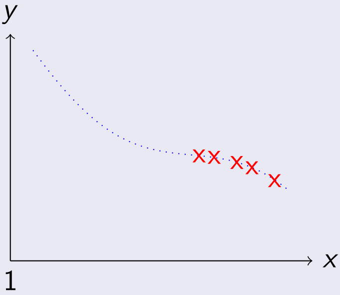
K representative paths

- heuristics,
- approximation,

Summarize Pareto Sets

How to summarize?

- **K-paths: output only K paths from the Pareto Set**



K representative paths

- heuristics,
- approximation,

Summarize Pareto Sets

How to summarize?

- K-paths,
- **heuristics: several metrics to evaluate the output quality**
- approximation,

Summarize Pareto Sets

How to summarize?

- K-paths,
- **heuristics: several metrics to evaluate the output quality**
  - [Hrnčíř et al., 2017] : pruning with different rules,
  - [Bast et al., 2013] : rounding and filtering,
  - [Delling et al., 2013] : MultiCriteria RAPTOR variants.
- approximation,

Summarize Pareto Sets

How to summarize?

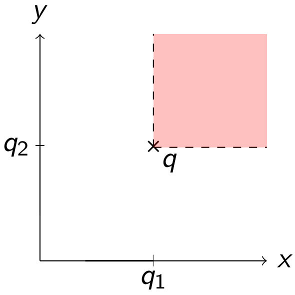
- K-paths,
- heuristics,
- **approximation:**

$(1+\epsilon)$-domination

Definition

$\mathbf{p} = (p_1, ...p_d)$ *is* $(1+\epsilon)$*-dominated by* $\mathbf{q} = (q_1, ..., q_d)$ *if:*

$$\forall i, q_i \leq (1+\epsilon) \cdot p_i$$



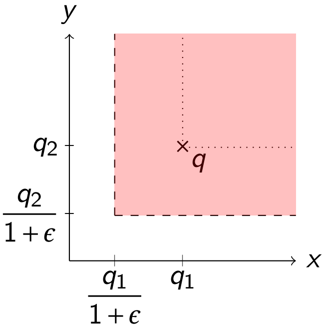
$(1+\epsilon)$ domination area

$(1+\epsilon)$-domination

Definition

$\mathbf{p} = (p_1, ...p_d)$ *is* $(1+\epsilon)$*-dominated by* $\mathbf{q} = (q_1, ..., q_d)$ *if:*

$$\forall i, q_i \leq (1+\epsilon) \cdot p_i$$



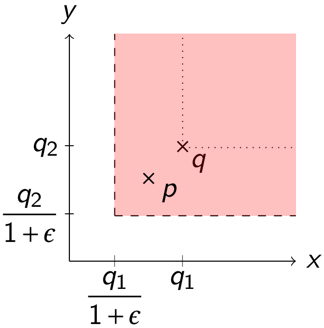
$(1+\epsilon)$ domination area

$(1+\epsilon)$-domination

Definition

$\mathbf{p} = (p_1, ...p_d)$ *is* $(1+\epsilon)$*-dominated by* $\mathbf{q} = (q_1, ..., q_d)$ *if:*

$$\forall i, q_i \leq (1+\epsilon) \cdot p_i$$

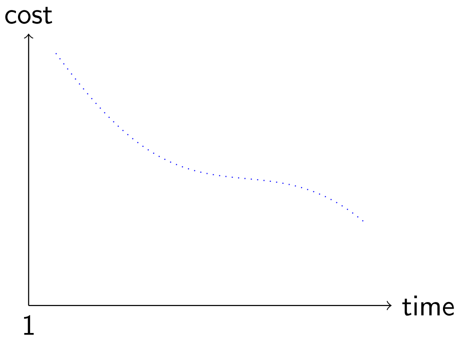


$(1+\epsilon)$ domination area

Approximation of the Pareto Set

Definition ($(1+\epsilon)$ approximation of a Pareto Set)

Set of paths S s.t. each path of the Pareto Set is $(1+\epsilon)$-dominated by one from S.

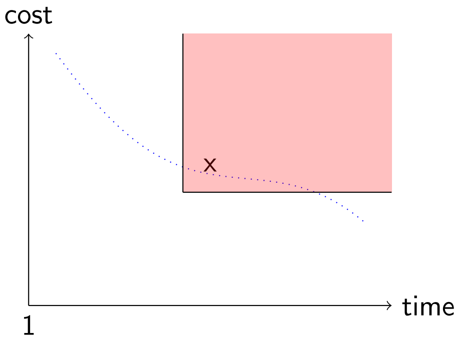


Approximation set with two dimensions

Approximation of the Pareto Set

Definition ($(1+\epsilon)$ approximation of a Pareto Set)

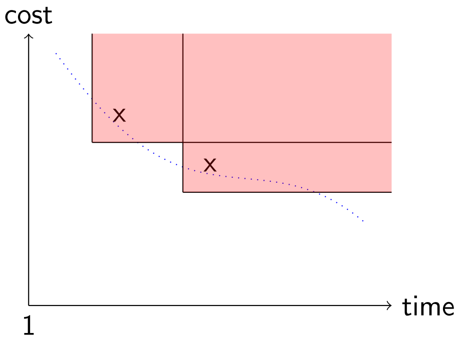
Set of paths S s.t. each path of the Pareto Set is $(1+\epsilon)$-dominated by one from S.



Approximation set with two dimensions

Definition ($(1+\epsilon)$ approximation of a Pareto Set)

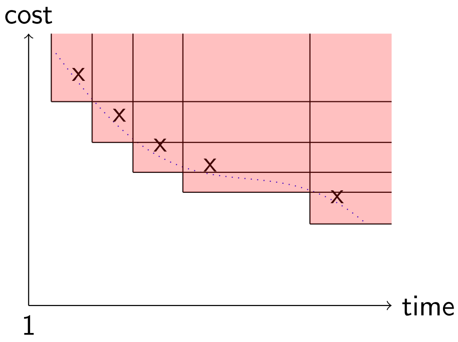
Set of paths S s.t. each path of the Pareto Set is $(1+\epsilon)$-dominated by one from S.



Approximation set with two dimensions

Definition ($(1+\epsilon)$ approximation of a Pareto Set)

Set of paths S s.t. each path of the Pareto Set is $(1+\epsilon)$-dominated by one from S.



Approximation set with two dimensions

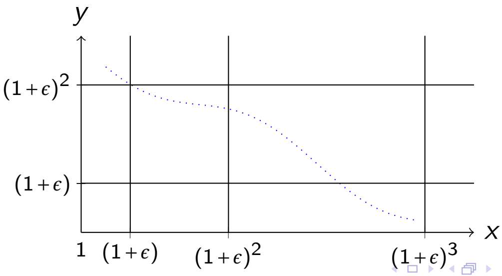
Polynomial cover set

Theorem

A polynomial approximation of a Pareto set exists [Papadimitriou and Yannakakis, 2000]:

$$|ApproxSet| = \mathcal{O}\left(\frac{\log(nW^{max})^{d-1}}{\epsilon^{d-1}}\right)$$

with W_{max} the maximal weight.



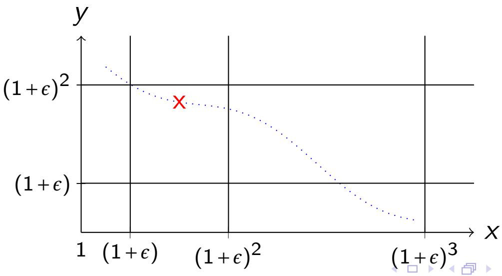
Polynomial cover set

Theorem

A polynomial approximation of a Pareto set exists [Papadimitriou and Yannakakis, 2000]:

$$|ApproxSet| = \mathcal{O}\left(\frac{\log(nW^{max})^{d-1}}{\epsilon^{d-1}}\right)$$

with W_{max} the maximal weight.

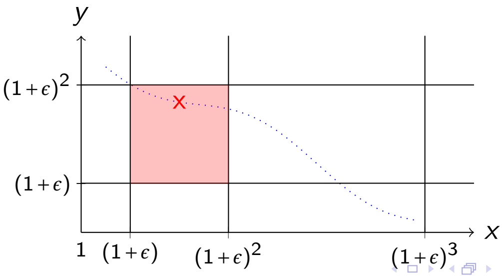


Theorem

A polynomial approximation of a Pareto set exists [Papadimitriou and Yannakakis, 2000]:

$$|ApproxSet| = \mathcal{O}\left(\frac{\log(nW^{max})^{d-1}}{\epsilon^{d-1}}\right)$$

with W_{max} the maximal weight.



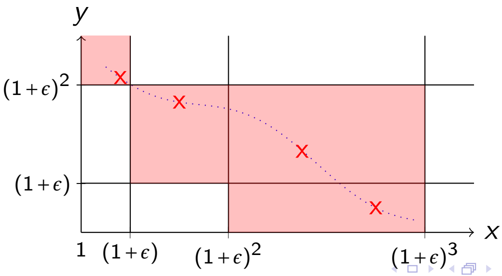
Polynomial cover set

Theorem

A polynomial approximation of a Pareto set exists [Papadimitriou and Yannakakis, 2000]:

$$|ApproxSet| = \mathcal{O}\left(\frac{\log(nW^{max})^{d-1}}{\epsilon^{d-1}}\right)$$

with W_{max} the maximal weight.



Polynomial approximation algorithm

Best known algorithm [Tsaggouris and Zaroliagis, 2009]

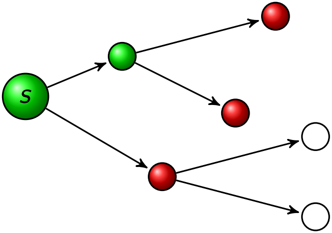
Bellman-Ford like algorithm with the following complexity:

$$\mathcal{O}\left(n^{d-1} \cdot n \cdot m \left(\frac{\log(nW^{max})}{\epsilon} \right)^{d-1} \right)$$

Problem: the factor n^{d-1} is unusable for large graphs.

Dijkstra-like algorithm

1. Priority queue containing paths ordered by the rank of the associated weights : $rank(p_1, \dots p_d) = p_1 + \dots + p_d$
2. We remove the minimum of the priority list and add it to the S_u.
3. We extend with all possible edges and add it to the priority queue.

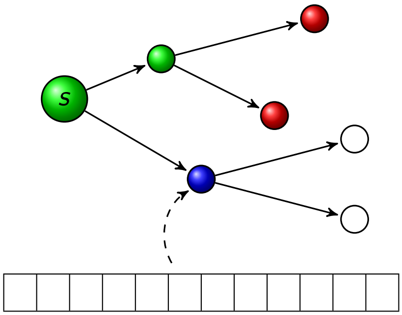


Priority queue

Exact algorithm (1)

Dijkstra-like algorithm

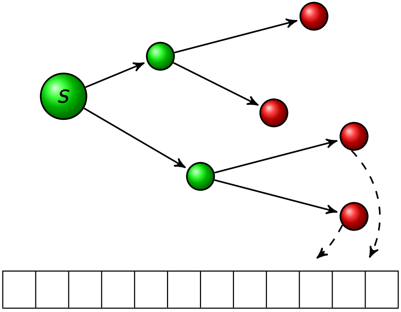
1. Priority queue containing paths ordered by the rank of the associated weights : $rank(p_1, ...p_d) = p_1 + ... + p_d$
2. We remove the minimum of the priority list and add it to the S_u.
3. We extend with all possible edges and add it to the priority queue.



Priority queue

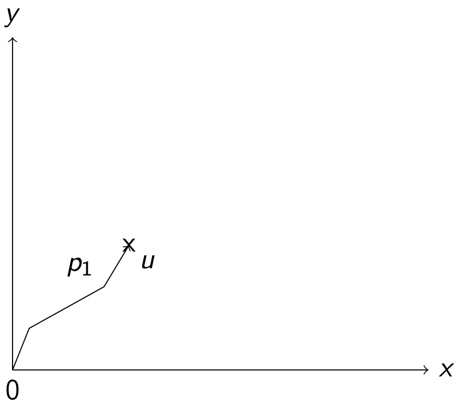
Dijkstra-like algorithm

1. Priority queue containing paths ordered by the rank of the associated weights : $rank(p_1, ...p_d) = p_1 + ... + p_d$
2. We remove the minimum of the priority list and add it to the S_u.
3. We extend with all possible edges and add it to the priority queue.



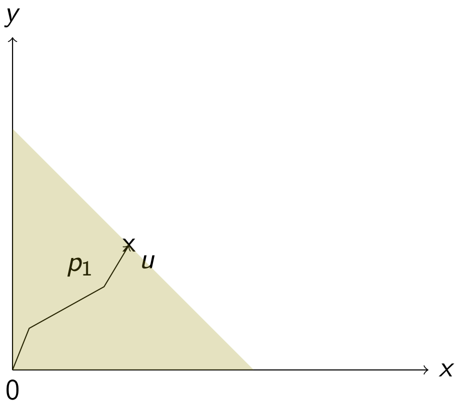
Priority queue

Exact algorithm (2)



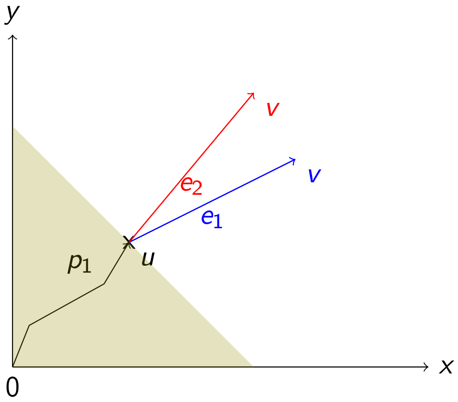
Extending paths

Exact algorithm (2)



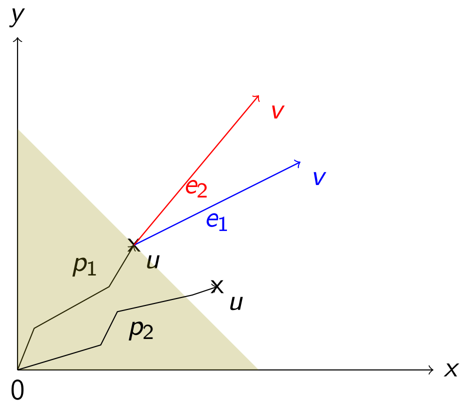
Extending paths

Exact algorithm (2)



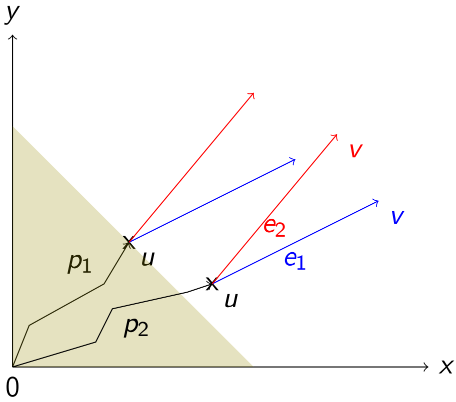
Extending paths

Exact algorithm (2)



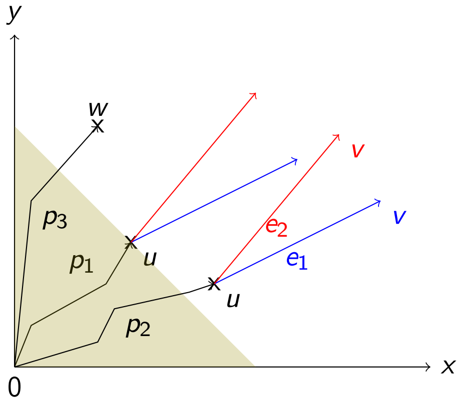
Extending paths

Exact algorithm (2)



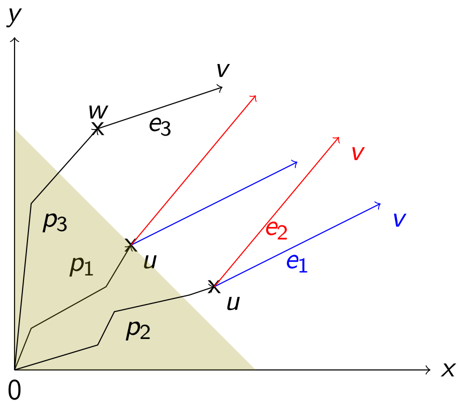
Extending paths

Exact algorithm (2)



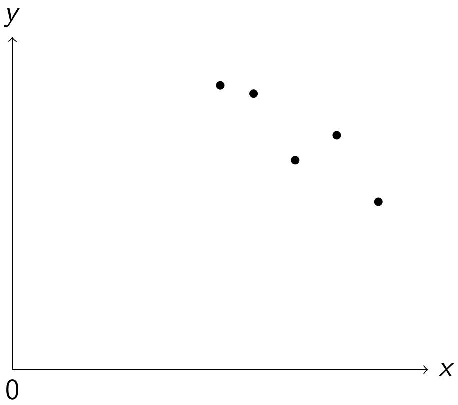
Extending paths

Exact algorithm (2)



Extending paths

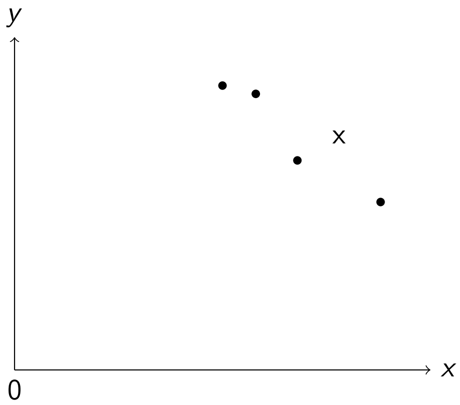
Exact algorithm (2)



Extending paths

We do not extend a path strictly dominated.

Exact algorithm (2)

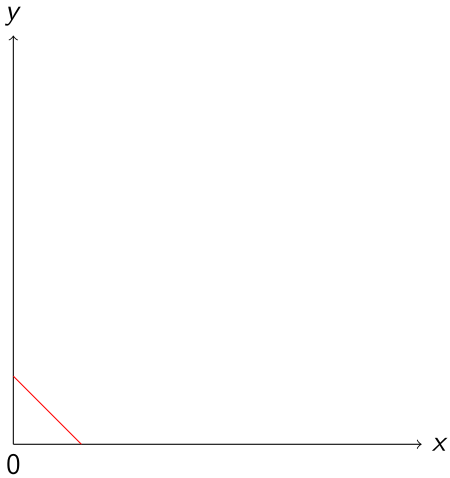


Extending paths

We do not extend a path strictly dominated.

Frame algorithm (1)

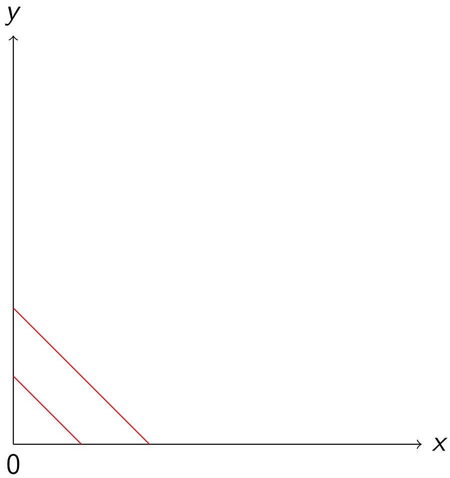
We firstly restrict to $d = 2$ and weight are in $\mathbb{N}^*$.



Frame algorithm

Frame algorithm (1)

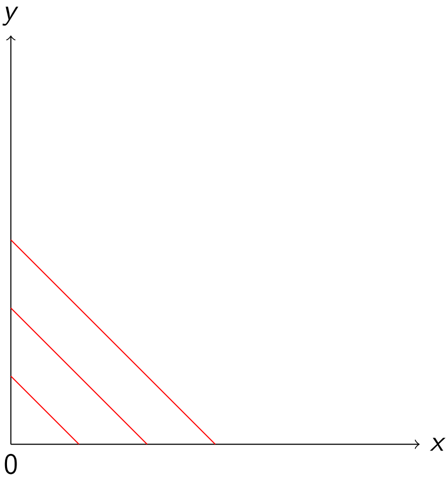
We firstly restrict to $d = 2$ and weight are in $\mathbb{N}^*$.



Frame algorithm

Frame algorithm (1)

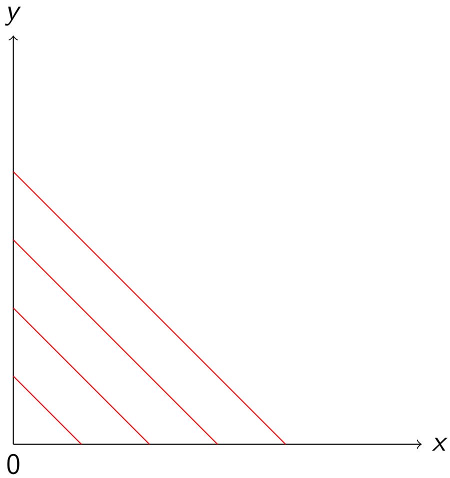
We firstly restrict to $d = 2$ and weight are in $\mathbb{N}^*$.



Frame algorithm

Frame algorithm (1)

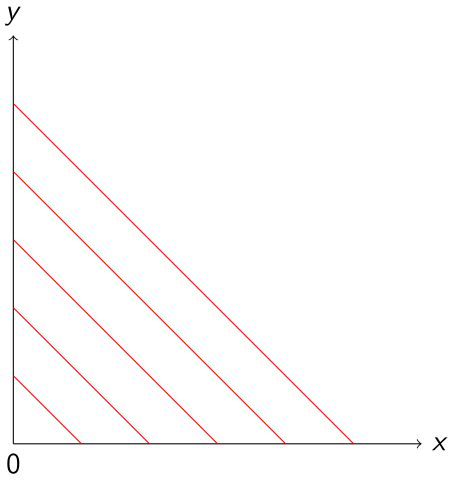
We firstly restrict to $d = 2$ and weight are in $\mathbb{N}^*$.



Frame algorithm

Frame algorithm (1)

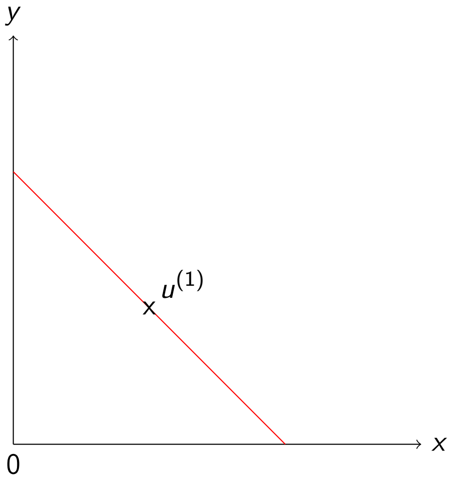
We firstly restrict to $d = 2$ and weight are in $\mathbb{N}^*$.



Frame algorithm

Frame algorithm (1)

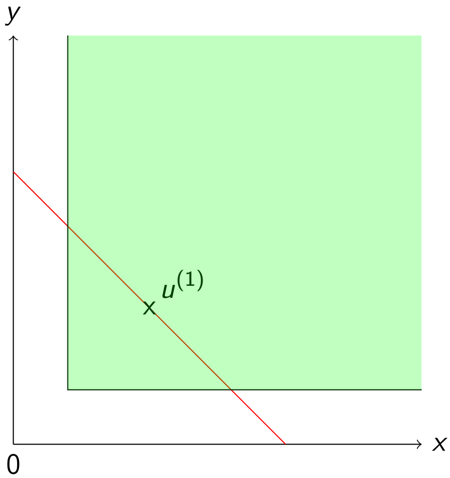
We firstly restrict to $d = 2$ and weight are in $\mathbb{N}^*$.



Frame algorithm

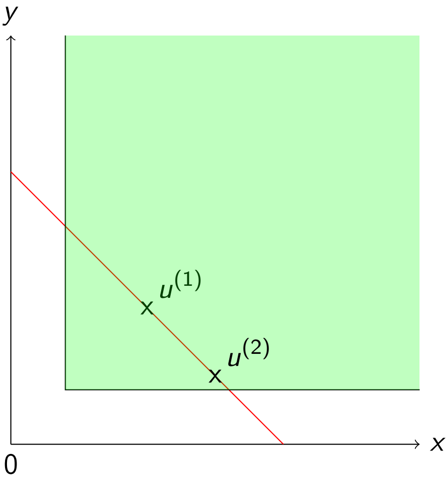
Frame algorithm (1)

We firstly restrict to $d = 2$ and weight are in $\mathbb{N}^*$.



Frame algorithm

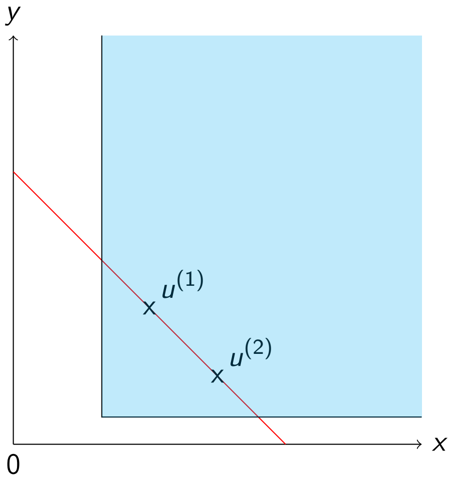
We firstly restrict to $d = 2$ and weight are in $\mathbb{N}^*$.



Frame algorithm

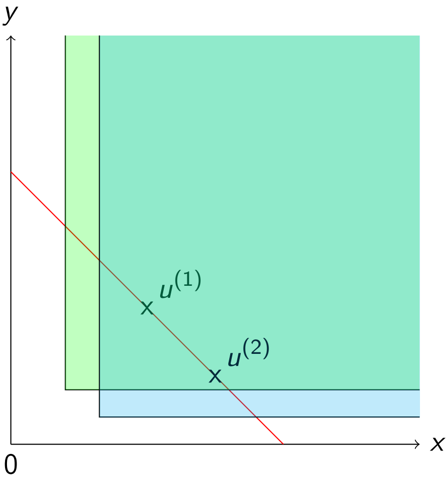
Frame algorithm (1)

We firstly restrict to $d = 2$ and weight are in $\mathbb{N}^*$.



Frame algorithm

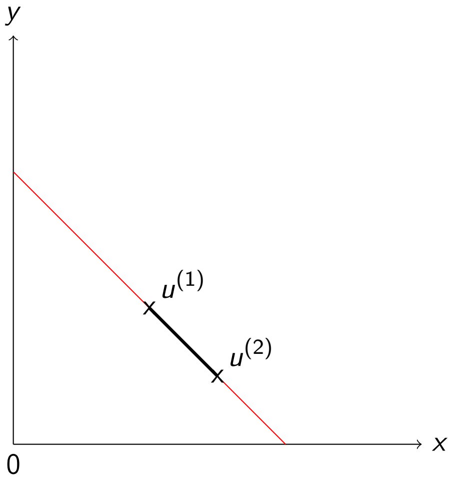
We firstly restrict to $d = 2$ and weight are in $\mathbb{N}^*$.



Frame algorithm

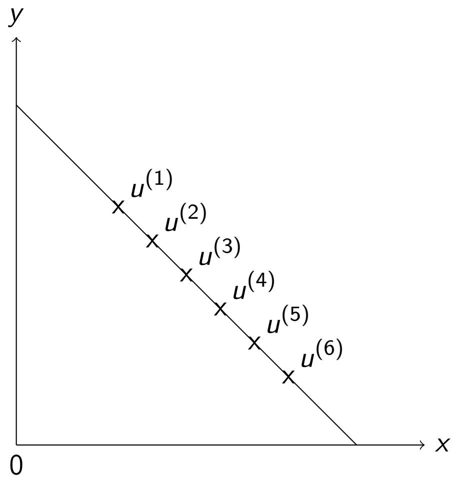
Frame algorithm (1)

We firstly restrict to $d = 2$ and weight are in $\mathbb{N}^*$.



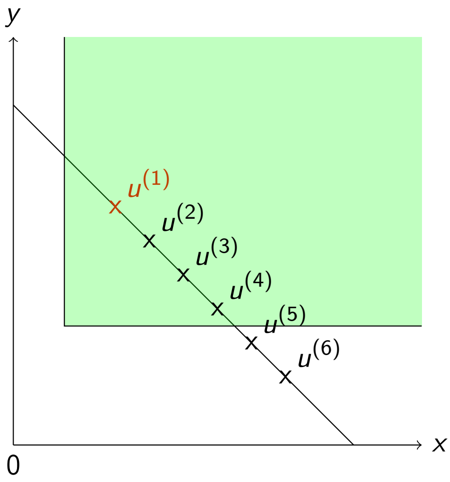
Frame algorithm

Frame algorithm (2)



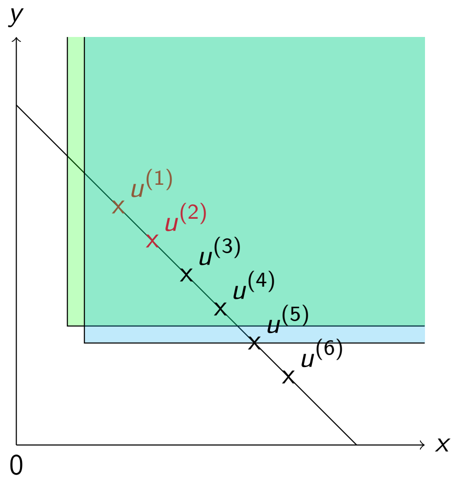
Frame algorithm

Frame algorithm (2)



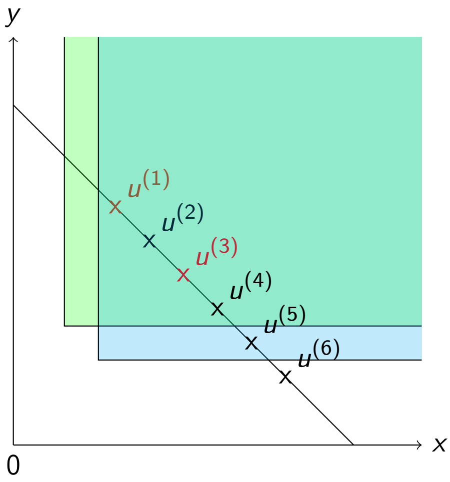
Frame algorithm

Frame algorithm (2)



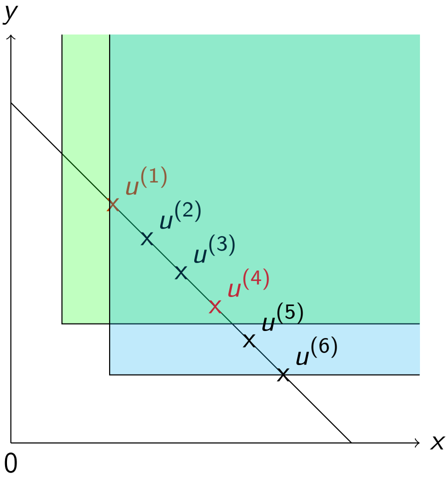
Frame algorithm

Frame algorithm (2)



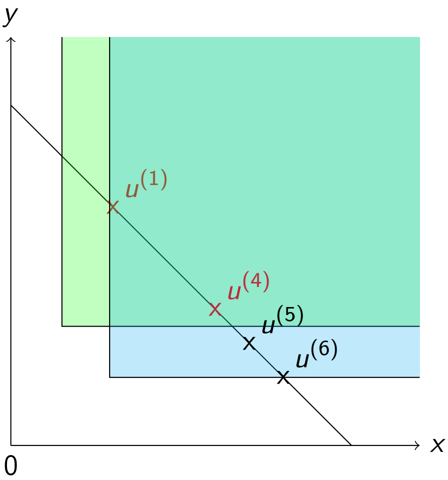
Frame algorithm

Frame algorithm (2)



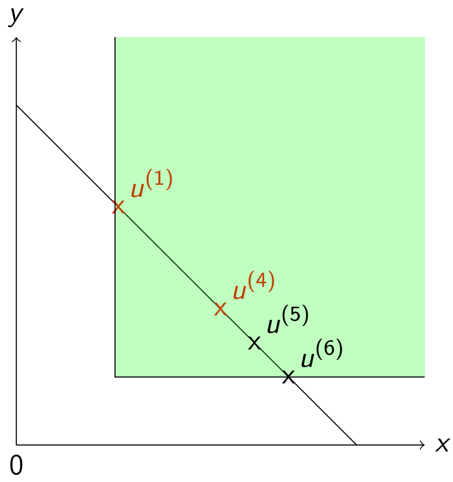
Frame algorithm

Frame algorithm (2)

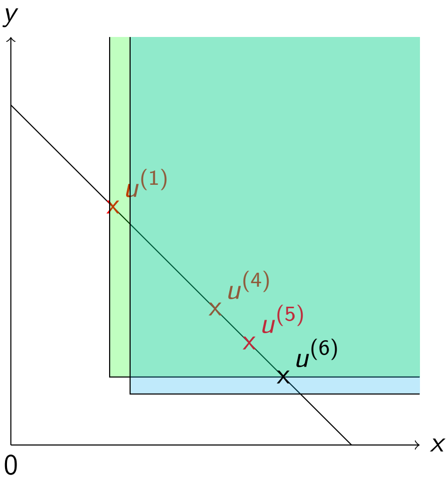


Frame algorithm

Frame algorithm (2)



Frame algorithm



Frame algorithm

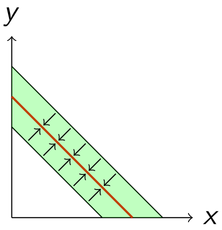
Frame algorithm (3)

Complexity (2D, integers weights)

Our Algorithm	$\mathcal{O}\left(\Delta \cdot W^{max} \cdot n \cdot \frac{n\log(nW^{max})}{\epsilon}\right)$
[Tsaggouris et al., 2009]	$\mathcal{O}\left(m \cdot n \cdot \frac{n\log(nW^{max})}{\epsilon}\right)$
[Vassilvitskii et al., 2005]	$\mathcal{O}\left(m \cdot (\log(log(n)) + \frac{1}{\epsilon}) \cdot \frac{n\log(nW^{max})}{\epsilon}\right)$

with Δ maximum degree and W^{max} maximum weight.

Weights in $\mathbb{R}_+^*$



Rounding inside a slice

Complexity

If the slice width is L, the algorithm gives an $(1+\epsilon+\frac{L}{2})$-approximation in

$$\mathcal{O}\left(\Delta \cdot W^{max} \cdot \frac{n^2 \log(nW^{max})}{L \cdot \epsilon}\right)$$

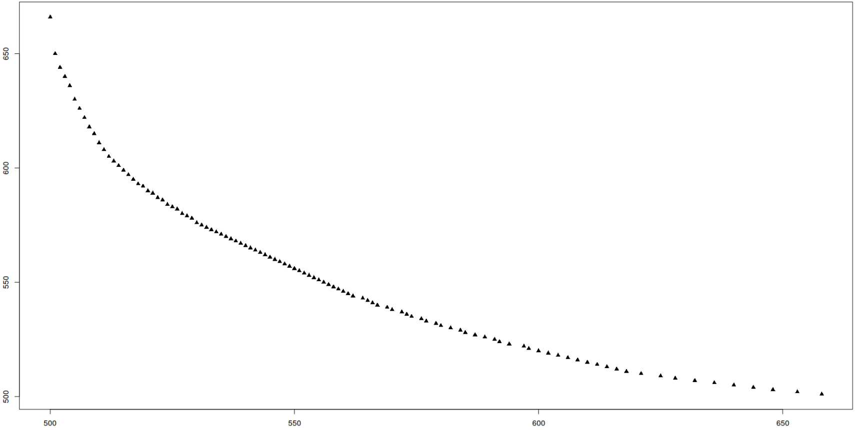
Weights in $\mathbb{R}_+^*$

Complexity (2D, non integer weights)

Frame Algorithm	$\mathcal{O}\left(\Delta \cdot W^{max} \cdot \frac{n^2 \log(nW^{max})}{\epsilon^2}\right)$
[Tsaggouris et al., 2009]	$\mathcal{O}\left(m \cdot \frac{n^2 \log(nW^{max})}{\epsilon}\right)$

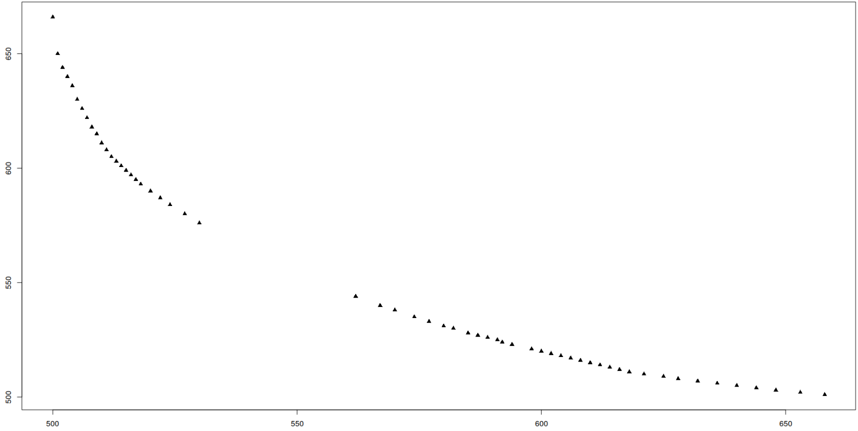
with Δ maximum degree and W^{max} maximum weight.

Tests



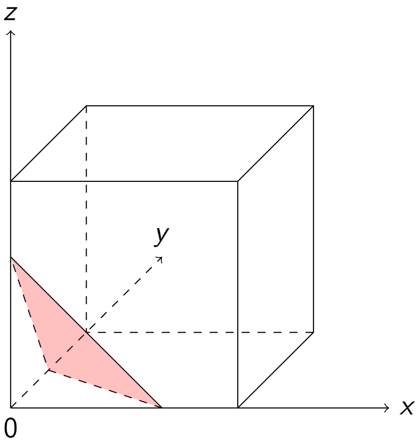
Pareto Set and Frame algorithm output

Tests



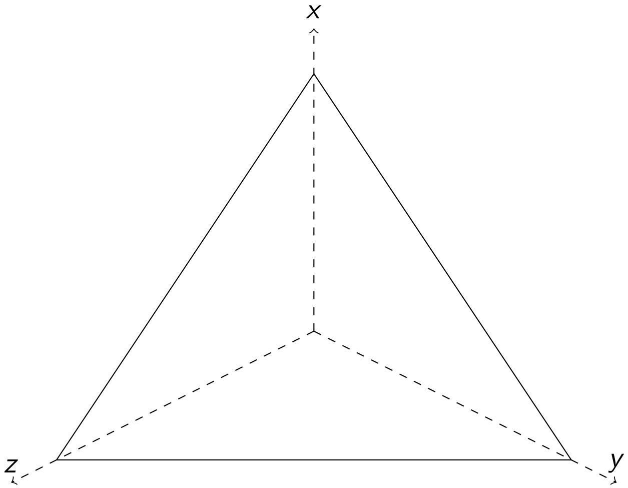
Pareto Set and Frame algorithm output

Higher dimensions (1)



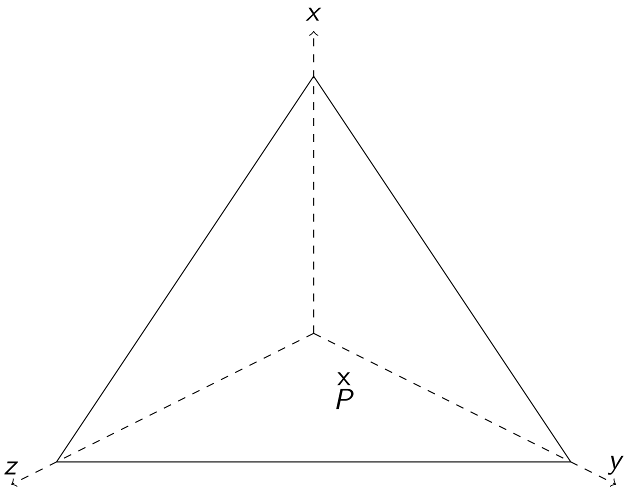
Same rank plan for $d = 3$

Higher dimensions (2)



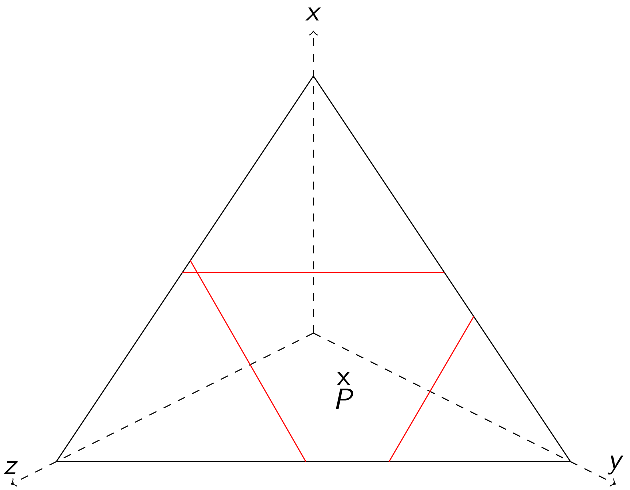
Elimination criterion

Higher dimensions (2)



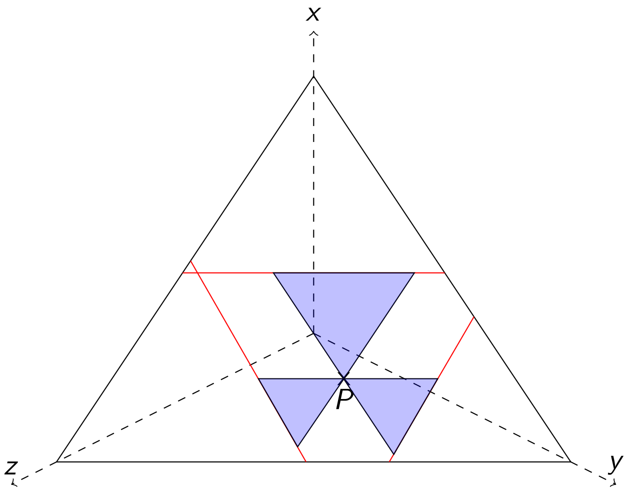
Elimination criterion

Higher dimensions (2)



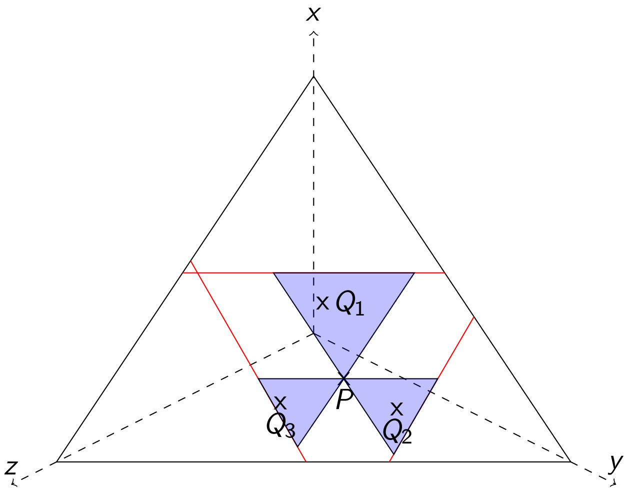
Elimination criterion

Higher dimensions (2)



Elimination criterion

Higher dimensions (2)



Elimination criterion

Higher dimensions (3)

Elimination criterion

Let S a set of same ranked paths and $P = (P_1, \ldots P_d) \in S$.
If $\forall i, \exists Q = (Q_1, \ldots, Q_d)$ such that

$$\begin{cases} P_i \leq Q_i \\ Q_i \leq (1+\epsilon)P_i \\ \forall j \neq i, P_j \geq Q_j \end{cases}$$

Then we remove P.

Lemme

If we apply the above criterion at each same rank hyperplan, the obtained solution set cover the Pareto Set.

Conclusion

Our algorithm

- we presented a new method to compute approximated Pareto Set,
- experiments on real datasets requiered (at least correlated weights),

Future work

A multimodal network enforces some generalisations:

- what about null weights?
- we work with static graph, what about temporal graph?
- can we efficiently use speed up techniques to improve query time ?

Conclusion

Our algorithm

- we presented a new method to compute approximated Pareto Set,
- experiments on real datasets requiered (at least correlated weights),

Future work

A multimodal network enforces some generalisations:

- what about null weights?
- we work with static graph, what about temporal graph?
- can we efficiently use speed up techniques to improve query time ?

Thanks for your attention !

[1] Leonid Antsfeld and Toby Walsh. Finding multi-criteria optimal paths in multi-modal public transportation networks using the transit algorithm. In *Proceedings of the 19th ITS World Congress*, page 32, 2012.

[2] Hannah Bast, Mirko Brodesser, and Sabine Storandt. Result diversity for multi-modal route planning. In *ATMOS-13th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems-2013*, volume 33, pages 123–136. Schloss Dagstuhl—Leibniz-Zentrum fuer Informatik, 2013.

[3] Hannah Bast, Daniel Delling, Andrew Goldberg, Matthias Müller-Hannemann, Thomas Pajor, Peter Sanders, Dorothea Wagner, and Renato F Werneck. Route planning in transportation networks. In *Algorithm engineering*, pages 19–80. Springer, 2016.

[4] Daniel Delling, Julian Dibbelt, Thomas Pajor, Dorothea Wagner, and Renato F Werneck. Computing multimodal journeys in practice. In *International Symposium on Experimental Algorithms*, pages 260–271. Springer, 2013.

[5] Daniel Delling, Thomas Pajor, and Renato F Werneck. Round-based public transit routing. *Transportation Science*, 49(3):591–604, 2014.

[6] Jan Hrnčíř, Pavol Žilecký, Qing Song, and Michal Jakob. Practical multicriteria urban bicycle routing. *IEEE Transactions on Intelligent Transportation Systems*, 18(3):493–504, 2017.

[7] Christos H Papadimitriou and Mihalis Yannakakis. On the approximability of trade-offs and optimal access of web sources. In *Foundations of Computer Science, 2000. Proceedings. 41st Annual Symposium on*, pages 86–92. IEEE, 2000.

[8] George Tsaggouris and Christos Zaroliagis. Multiobjective optimization: Improved fptas for shortest paths and non-linear objectives with applications. *Theory of Computing Systems*, 45(1):162–186, 2009.

[9] Dorothea Wagner and Tobias Zündorf. Public transit routing with unrestricted walking. In *OASIcs-OpenAccess Series in Informatics*, volume 59. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.