On two speeding-up techniques for the computation of multi-objective shortest paths with a label setting algorithm

RAMOO'2018: 5th International Workshop on Recent Advances in Multi-Objective Optimization
November 15, 2018 – Université de Nantes, France

Nicolas FORGET[1]
with
Xavier GANDIBLEUX[1], Didier ROBBES[1], Matthias EHRGOTT[2]

1: Université de Nantes (France)
2: University of Lancaster (United Kingdom)

Supported by

vOpt Research Project

Exact Efficient Solution of Mixed Integer Programming Problems with Multiple Objective Functions (ANR/DFG-14-CE35-0034-01)

1. Introduction

Assumptions and notations

- On the problem:
  - $G(N, A, C)$, a static, connected, directed and valued graph
  - $p - \sum$, i.e. p linear objectives with $p \geqslant 2$
  - Compute 1-to-all (source s to other nodes) shortest paths
- On the instance:
  - No specific topology for G
  - p positive costs on arcs
- On the solutions:
  - No preference on shortest paths
  - $s \in N$ given:
    efficient paths over p objectives from s to all $t \in N \backslash \{s\}$,
    X_E, a complete set of efficient paths,
    $Y_N = Z(X_E)$, the set of non-dominated points
- On the algorithm:
  - Label setting principle
  - Martins' algorithm (1984)

Assumptions and notations

- On the problem:
  - $G(N, A, C)$, a static, connected, directed and valued graph
  - $p - \sum$, i.e. p linear objectives with $p \geqslant 2$
  - Compute 1-to-all (source s to other nodes) shortest paths
- On the instance:
  - No specific topology for G
  - p positive costs on arcs
- On the solutions:
  - No preference on shortest paths
  - $s \in N$ given:
    efficient paths over p objectives from s to all $t \in N \backslash \{s\}$,
    X_E, a complete set of efficient paths,
    $Y_N = Z(X_E)$, the set of non-dominated points
- On the algorithm:
  - Label setting principle
  - Martins' algorithm (1984)

Assumptions and notations

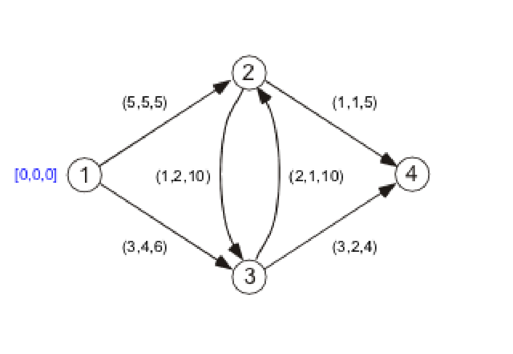
- On the problem:
  - $G(N, A, C)$, a static, connected, directed and valued graph
  - $p - \sum$, i.e. p linear objectives with $p \geqslant 2$
  - Compute 1-to-all (source s to other nodes) shortest paths
- On the instance:
  - No specific topology for G
  - p positive costs on arcs
- On the solutions:
  - No preference on shortest paths
  - $s \in N$ given:
    efficient paths over p objectives from s to all $t \in N \backslash \{s\}$,
    X_E, a complete set of efficient paths,
    $Y_N = Z(X_E)$, the set of non-dominated points
- On the algorithm:
  - Label setting principle
  - Martins' algorithm (1984)

Assumptions and notations

- On the problem:
  - $G(N, A, C)$, a static, connected, directed and valued graph
  - $p - \sum$, i.e. p linear objectives with $p \geqslant 2$
  - Compute 1-to-all (source s to other nodes) shortest paths
- On the instance:
  - No specific topology for G
  - p positive costs on arcs
- On the solutions:
  - No preference on shortest paths
  - $s \in N$ given:
    efficient paths over p objectives from s to all $t \in N \backslash \{s\}$,
    X_E, a complete set of efficient paths,
    $Y_N = Z(X_E)$, the set of non-dominated points
- On the algorithm:
  - Label setting principle
  - Martins' algorithm (1984)

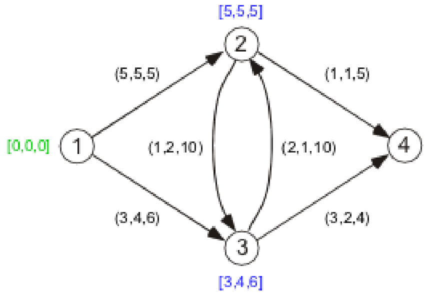
Ingredients and principle of Martins' algorithm

- Temporary and permanent labels
- Lexicographic selection of a temporary label
- Propagation principle over outgoing arcs
- All permanent labels correspond to efficient paths
- Pruning temporary labels on nodes by dominance



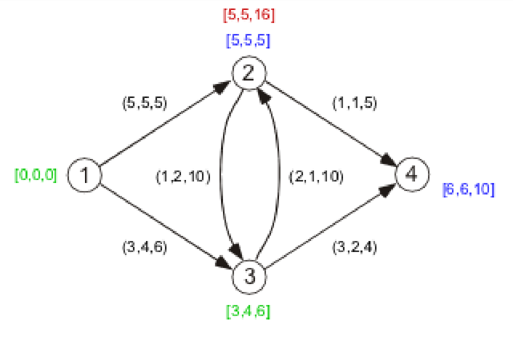
Ingredients and principle of Martins' algorithm

- Temporary and permanent labels
- Lexicographic selection of a temporary label
- Propagation principle over outgoing arcs
- All permanent labels correspond to efficient paths
- Pruning temporary labels on nodes by dominance



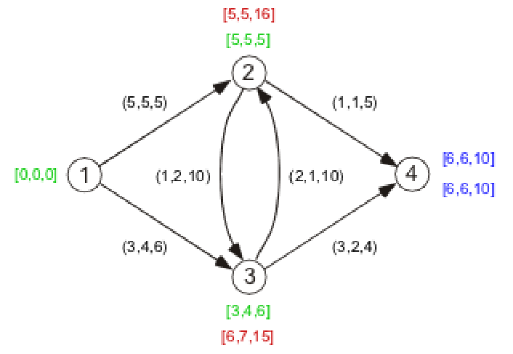
Ingredients and principle of Martins' algorithm

- Temporary and permanent labels
- Lexicographic selection of a temporary label
- Propagation principle over outgoing arcs
- All permanent labels correspond to efficient paths
- Pruning temporary labels on nodes by dominance



Ingredients and principle of Martins' algorithm

- Temporary and permanent labels
- Lexicographic selection of a temporary label
- Propagation principle over outgoing arcs
- All permanent labels correspond to efficient paths
- Pruning temporary labels on nodes by dominance



Motivation and Questions

- Motivations: good base
    - to investigate the influence of an instance on the algorithm
    - to measure the impact of additional components on the algorithm
    - to develop an implementation to be integrated into vOptSolver

- Questions:
    - Concerning the maintenance of non-dominated temporary labels (operations of comparison, insertion, deletion) on nodes:

      What is the added value of an advanced data structure for maintaining on nodes during the iterations of the algorithm?

    - Concerning the generation of temporary labels on nodes:

      What is the added value of a two-directional strategy on the total number of temporary labels generated by the algorithm?

Motivation and Questions

- Motivations: good base
  - to investigate the influence of an instance on the algorithm
  - to measure the impact of additional components on the algorithm
  - to develop an implementation to be integrated into vOptSolver

- Questions:
  - Concerning the maintenance of non-dominated temporary labels (operations of comparison, insertion, deletion) on nodes:

    What is the added value of an advanced data structure for maintaining on nodes during the iterations of the algorithm?

  - Concerning the generation of temporary labels on nodes:

    What is the added value of a two-directional strategy on the total number of temporary labels generated by the algorithm?

Motivation and Questions

- Motivations: good base
  - to investigate the influence of an instance on the algorithm
  - to measure the impact of additional components on the algorithm
  - to develop an implementation to be integrated into vOptSolver

- Questions:
  - Concerning the maintenance of non-dominated temporary labels (operations of comparison, insertion, deletion) on nodes:

    What is the added value of an advanced data structure for maintaining on nodes during the iterations of the algorithm?

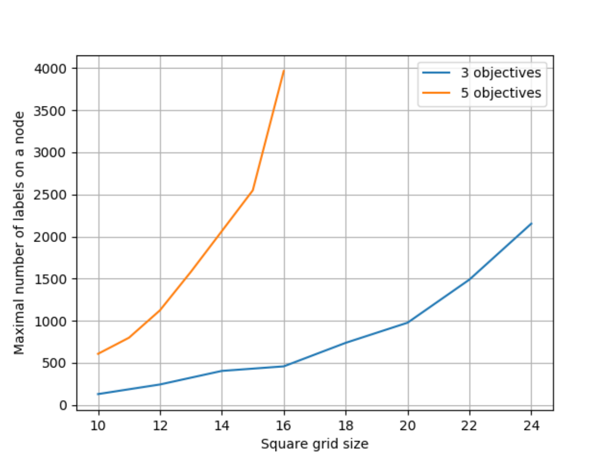
  - Concerning the generation of temporary labels on nodes:

    What is the added value of a two-directional strategy on the total number of temporary labels generated by the algorithm?

2. On the strategy to maintain temporary labels

Observation

- Maximal number of labels (temporary and permanents) on 1 node:



Situation

- Maintaining non-dominated labels on nodes requires operations of comparison, insertion, deletion.
- A multi-dimensional data structure is then required for storing the information and facilitating these operations during the iterations of the algorithm.
- The data structures found in specialized literature in MOO are:
  - a linear structure
    list, sorted or not
    $\rightarrow$ simple vs cost of the pairwise comparison
  - a tree structure
    AVL-Tree ($p = 2$), Quad-Tree ($p \geqslant 2$), ND-Tree ($p \geqslant 2$)
    $\rightarrow$ fast vs complexity for maintaining the structure

Situation

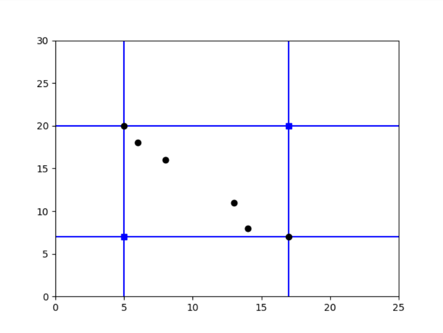
- Maintaining non-dominated labels on nodes requires operations of comparison, insertion, deletion.
- A multi-dimensional data structure is then required for storing the information and facilitating these operations during the iterations of the algorithm.
- The data structures found in specialized literature in MOO are:
  - a linear structure
    list, sorted or not
    $\rightarrow$ simple vs cost of the pairwise comparison
  - a tree structure
    AVL-Tree ($p = 2$), Quad-Tree ($p \geqslant 2$), ND-Tree ($p \geqslant 2$)
    $\rightarrow$ fast vs complexity for maintaining the structure

Situation

- Maintaining non-dominated labels on nodes requires operations of comparison, insertion, deletion.
- A multi-dimensional data structure is then required for storing the information and facilitating these operations during the iterations of the algorithm.
- The data structures found in specialized literature in MOO are:
  - a linear structure
    list, sorted or not
    $\rightarrow$ simple vs cost of the pairwise comparison
  - a tree structure
    AVL-Tree ($p = 2$), Quad-Tree ($p \geqslant 2$), ND-Tree ($p \geqslant 2$)
    $\rightarrow$ fast vs complexity for maintaining the structure

ND-Tree

Andrzej Jaszkiewicz and Thibaut Lust, "ND-Tree-Based Update: A Fast Algorithm for the Dynamic Nondominance Problem". *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 5, pp. 778-791, Oct. 2018.



Principle:

- to divide the objective space into hyperrectangles
- if a hyperrectangle contains too many points then it is divided

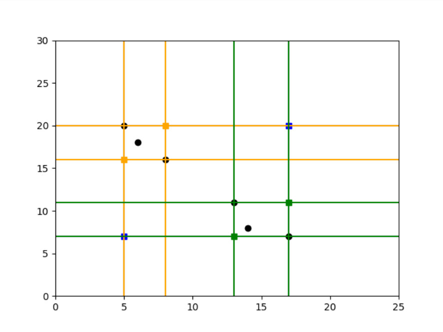
Parameters:

σ: maximal number of points per hyperrectangle

δ: number of sub-hyperrectangles created when a hyperrectangle is divided

ND-Tree

Andrzej Jaszkiewicz and Thibaut Lust, “ND-Tree-Based Update: A Fast Algorithm for the Dynamic Nondominance Problem”. *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 5, pp. 778-791, Oct. 2018.



Principle:

- to divide the objective space into hyperrectangles
- if a hyperrectangle contains too much points then it is divided

Parameters:

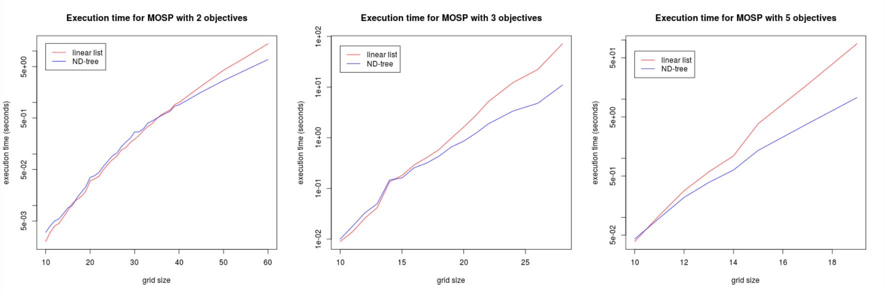
σ: maximal number of points per hyperrectangle

δ: number of sub-hyperrectangles created when a hyperrectangle is divided

ND-Tree: numerical experiments

- One ND-Tree on each node of the graph:

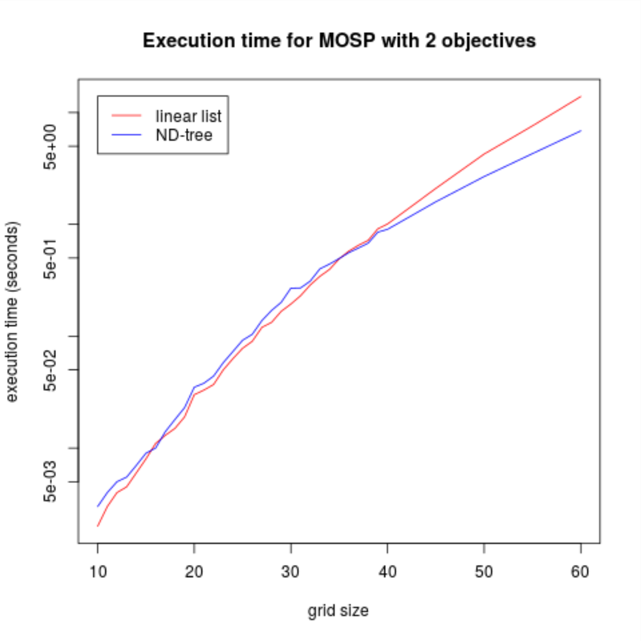
Linear list vs ND-tree for a grid graph. Cost values are in $U[1; 100]$



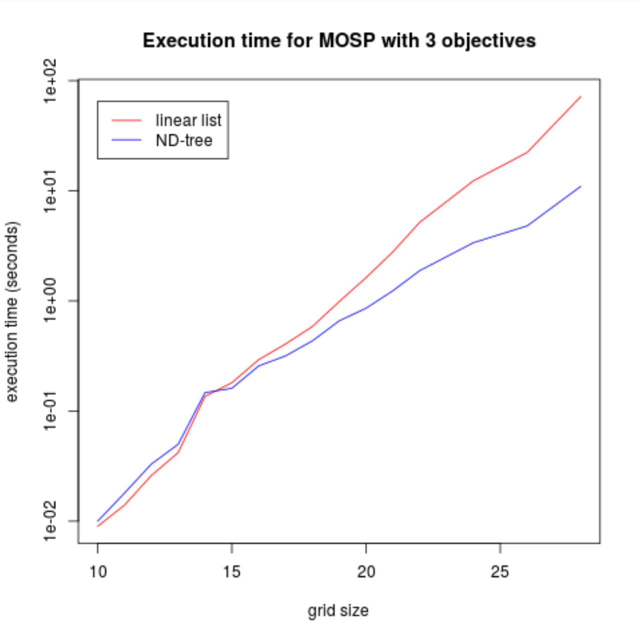
X axis: Number of nodes (gridSize × gridsize) — Y axis: CPUt (in seconds) in logarithmic scale — Results in average on 50 runs

⇒ ND-tree appears competitive against the linear list when the number of objectives and the number of nodes are high

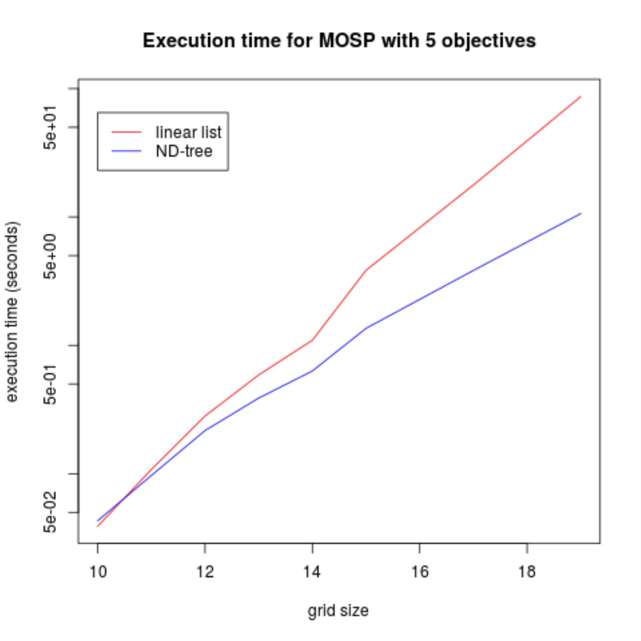
▸▸ goto next section



X axis: Number of nodes (gridSize × gridsize) – Y axis: CPU time (in seconds) in logarithmic scale – Results on average for 50 runs



X axis: Number of nodes (gridSize × gridsize) – Y axis: CPU time (in seconds) in logarithmic scale – Results on average for 50 runs

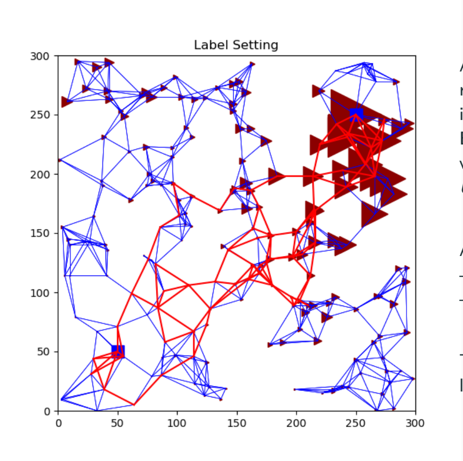


X axis: Number of nodes (gridSize $\times$ gridsize) – Y axis: CPU time (in seconds) in logarithmic scale – Results on average for 50 runs

3. On the strategy of label propagation

Observation

- New assumption:
  1-to-1 (source s to destination t) efficient paths



A random graph with clusters (200 nodes randomly generated, each node is connected with its 4 closest neighbors according to the e Euclidean distance) and 3 linear objectives. Cost values are randomly selected in the range $U[1;300]$

All efficient shortest paths (in red) between
- the origin node (square in the south-west) and
- the destination node (square in the north-east)

The triangles represent the maximum number of labels on a node

Situation

- Number of labels may grow significantly near the termination node t.

- Alternative: a bi-directional strategy
  - Well-known in single objective case (Nicholson 1966; Pohl 1969)
  - Use two separated procedures:
    - a forward search from the origin node and
    - a backward search starting from the destination node

      $\rightarrow$ two search trees,
      potentially expanding fewer labels than a single search

- Existing literature in MOP:
  - Demeyer et al., 2013 (4OR journal)
    $p \geqslant 2$
  - Galand et al., 2013 (SOCS'2013 conference)
    user preferences, prefered paths
  - Sedeño-Noda and Colebrook, 2018 (EURO'2018 conf.)
    $p = 2$

Situation

- Number of labels may grow significantly near the termination node t.

- Alternative: a bi-directional strategy
  - Well-known in single objective case (Nicholson 1966; Pohl 1969)
  - Use two separated procedures:
    - a forward search from the origin node and
    - a backward search starting from the destination node

      $\rightarrow$ two search trees,
      potentially expanding fewer labels than a single search

- Existing literature in MOP:
  - Demeyer et al., 2013 (4OR journal)
    $p \geqslant 2$
  - Galand et al., 2013 (SOCS'2013 conference)
    user preferences, prefered paths
  - Sedeño-Noda and Colebrook, 2018 (EURO'2018 conf.)
    $p = 2$

Situation

- Number of labels may grow significantly near the termination node t.
- Alternative: a bi-directional strategy
  - Well-known in single objective case (Nicholson 1966; Pohl 1969)
  - Use two separated procedures:
    - a forward search from the origin node and
    - a backward search starting from the destination node

      $\rightarrow$ two search trees,
      potentially expanding fewer labels than a single search
- Existing literature in MOP:
  - Demeyer et al., 2013 (4OR journal)
    $p \geqslant 2$
  - Galand et al., 2013 (SOCS'2013 conference)
    user preferences, prefered paths
  - Sedeño-Noda and Colebrook, 2018 (EURO'2018 conf.)
    $p = 2$

Speed-up techniques proposed in Demeyer et al., 2013

Main ideas:

- a forward and backward search (similar to the unidirectional algorithm)
- a stopping condition is based on the use of a vector of minimal values of objectives for temporary labels

Numerical experiments:

- up to 20 time faster for transportation graphs with 2 and 3 objectives (instances: sparse graphs representing transportation problems with hundreds of thousands nodes and links, average node degree between 2 and 3)
- mitigated for random graphs
- not pertinent for complete graphs and square grid graphs

⇒ performance of the strategy is dependant on the graph configuration and predicting the average speedup is difficult.

Speed-up techniques proposed in Demeyer et al., 2013

Main ideas:

- a forward and backward search (similar to the unidirectional algorithm)
- a stopping condition is based on the use of a vector of minimal values of objectives for temporary labels

Numerical experiments:

- up to 20 time faster for transportation graphs with 2 and 3 objectives (instances: sparse graphs representing transportation problems with hundreds of thousands nodes and links, average node degree between 2 and 3)
- mitigated for random graphs
- not pertinent for complete graphs and square grid graphs

⇒ performance of the strategy is dependant on the graph configuration and predicting the average speedup is difficult.

Speed-up techniques proposed in Demeyer et al., 2013

Main ideas:

- a forward and backward search (similar to the unidirectional algorithm)
- a stopping condition is based on the use of a vector of minimal values of objectives for temporary labels

Numerical experiments:

- up to 20 time faster for transportation graphs with 2 and 3 objectives (instances: sparse graphs representing transportation problems with hundreds of thousands nodes and links, average node degree between 2 and 3)
- mitigated for random graphs
- not pertinent for complete graphs and square grid graphs

$\Rightarrow$ performance of the strategy is dependant on the graph configuration and predicting the average speedup is difficult.

Our ongoing attempt on the general case

- Observation: good results appear with a specific graph topology or when not considering a subset of X_E.
- Question: can we find a way to improve the computation time using a bi-directional strategy considering:
  - a (minimal) complete set of X_E
  - any graph topology
- Proposals :
  - a separation of the graph in two sub-graphs in preprocessing
  - identify graph topologies where a bi-directional separation strategy may be interesting or unnecessary

Our ongoing attempt on the general case

- Observation: good results appear with a specific graph topology or when not considering a subset of X_E.

- Question: can we find a way to improve the computation time using a bi-directional strategy considering:
  - a (minimal) complete set of X_E
  - any graph topology

- Proposals :
  - a separation of the graph in two sub-graphs in preprocessing
  - identify graph topologies where a bi-directional separation strategy may be interesting or unnecessary

Our ongoing attempt on the general case

- Observation: good results appear with a specific graph topology or when not considering a subset of X_E.

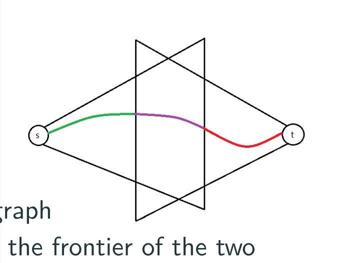
- Question: can we find a way to improve the computation time using a bi-directional strategy considering:
  - a (minimal) complete set of X_E
  - any graph topology

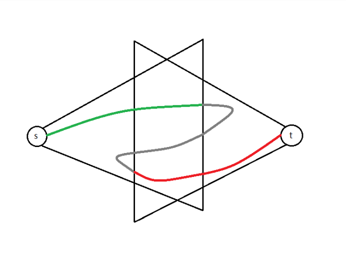
- Proposals :
  - a separation of the graph in two sub-graphs in preprocessing
  - identify graph topologies where a bi-directional separation strategy may be interesting or unnecessary

About the separation of the graph

Main ideas:

- compute a separation of the graph in two parts such that the source is in one sub-graph and the destination node is in the other one
- apply Martins' algorithm on each sub-graph
- merge the permanent labels existing at the frontier of the two sub-graphs





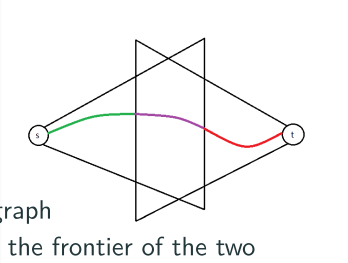
Difficulties :

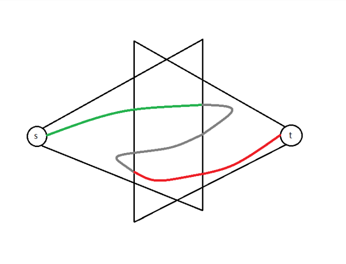
- cost of merging of the paths
- are we sure to compute Y_N?

About the separation of the graph

Main ideas:

- compute a separation of the graph in two parts such that the source is in one sub-graph and the destination node is in the other one
- apply Martins' algorithm on each sub-graph
- merge the permanent labels existing at the frontier of the two sub-graphs





Difficulties :

- cost of merging of the paths
- are we sure to compute Y_N?

About the separation of the graph

- Question: is it possible to find a way to separate the graph such that there is no paths crossing each sub-graph twice or more?
    - *no* in general in an undirected graph
    - *maybe* in a directed graph
    - *yes* in a "well-oriented" graph

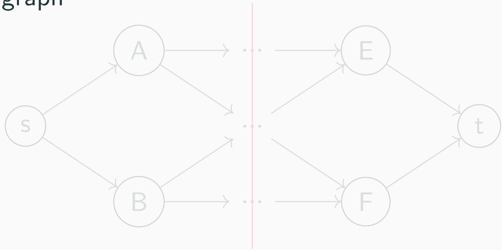
    - an example of "well-oriented" graph

- Remaining questions:
    - efficiency of a separation strategy in "well-oriented" graph?
    - can we find a way to compute a good separation in more random directed graphs?

About the separation of the graph

- Question: is it possible to find a way to separate the graph such that there is no paths crossing each sub-graph twice or more?
    - *no* in general in an undirected graph
    - *maybe* in a directed graph
    - *yes* in a "well-oriented" graph

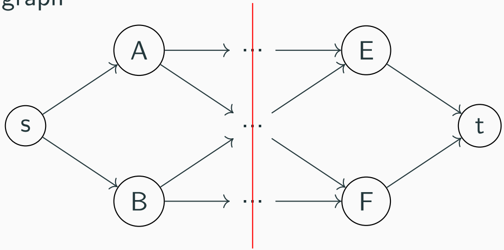
- an example of "well-oriented" graph



- Remaining questions:
    - efficiency of a separation strategy in "well-oriented" graph?
    - can we find a way to compute a good separation in more random directed graphs?

About the separation of the graph

- Question: is it possible to find a way to separate the graph such that there is no paths crossing each sub-graph twice or more?
  - *no* in general in an undirected graph
  - *maybe* in a directed graph
  - *yes* in a "well-oriented" graph
  - an example of "well-oriented" graph

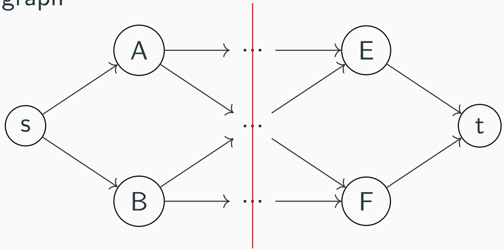


- Remaining questions:
  - efficiency of a separation strategy in "well-oriented" graph?
  - can we find a way to compute a good separation in more random directed graphs?

About the separation of the graph

- Question: is it possible to find a way to separate the graph such that there is no paths crossing each sub-graph twice or more?
  - *no* in general in an undirected graph
  - *maybe* in a directed graph
  - *yes* in a "well-oriented" graph

- an example of "well-oriented" graph



- Remaining questions:
  - efficiency of a separation strategy in "well-oriented" graph?
  - can we find a way to compute a good separation in more random directed graphs?

4. Summary

Conclusion and ongoing works

→ A novel context of using ND-tree for maintaining temporary labels

→ A learning on the bi-directional strategy

→ A (coming) open-source package dealing with several MOSP

PROS:

- ND-tree: interesting even with few objectives
- bi-directional strategy: interesting for transportation graphs

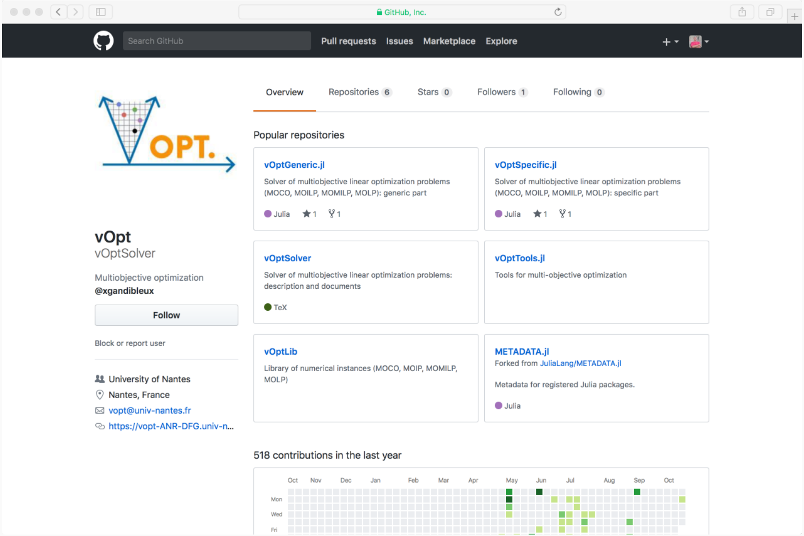
CONS:

- ND-tree: parameters to tune
- bi-directional strategy: predicting the average speedup is difficult

NOW:

- ND-tree: measuring the impact of σ and δ
- Bi-directional strategy: dealing with stated questions
- Label setting algorithm: working on others pending questions
- vOptSolver: releasing the `MOSP.jl` package

MOSP.jl awaited to be integrated to vOptSolver



http://voptsolver.github.io/vOptSolver/

References i

- Sofie Demeyer, Jan Goedgebeur, Pieter Audenaert, Mario Pickavet, and Piet Demeester, *Speeding up martins' algorithm for multiple objective shortest path problems*, 4OR **11** (2013), no. 4, 323–348.
- Nicolas Forget, *Plus courts chemins multi-objectifs*, Research project (year 1 of MSc in Computer Science track Optimization in Operations Research), Université de Nantes, May 2018, Defended the 30th of May 2018. In french.
- Lucie Galand, Anisse Ismaili, Patrice Perny, and Olivier Spanjaard, *Bidirectional preference-based search for state space graph problems*, Proceedings of the Sixth Annual Symposium on Combinatorial Search, SOCS 2013, Leavenworth, Washington, USA, July 11-13, 2013.

- Xavier Gandibleux, Gauthier Soleilhac, Anthony Przybylski, and Stefan Ruzika, *vOptSolver: an open source software environment for multiobjective mathematical optimization*, IFORS2017: 21st Conference of the International Federation of Operational Research Societies. Quebec City (Canada), 2017, July 17-21.
- Andrzej Jaszkiewicz and Thibaut Lust, *ND-Tree-based update: a fast algorithm for the dynamic non-dominance problem*, IEEE Transactions on Evolutionary Computation **22** (2018), no. 5, 778–791.
- Ernesto Queirós Vieira Martins, *On a multicriteria shortest path problem*, European Journal of Operational Research **16** (1984), no. 2, 236 – 245.
- Ira Pohl, *Bi-directional and heuristic search in path problems*, 1969, Stanford University – SLAC-104 UC-32 (MISC).

References iii

- Antonio Sedeño-Noda and Marcos Colebrook, *The multiobjective Dijkstra's algorithm*, 2018, EURO'2018, July 8-11, 2018. Valencia, Spain.
- vOptSolver, *Homepage of voptsolver*, 2017, `http://voptsolver.github.io/vOptSolver/`. Last update: Oct 2018.

On two speeding-up techniques for the computation of multi-objective shortest paths with a label setting algorithm

Nicolas Forget, Xavier Gandibleux, Didier Robbes, Matthias Ehrgott

Homepage of vOptSolver:
`http://voptsolver.github.io/vOptSolver/`

Repository of vOptSolver:
`http://github.com/vOptSolver`

Contact concerning vOptSolver:
`vopt@univ-nantes.fr`

Follow vOptSolver on Twitter:
`@vOptSolver`