Heuristic and exact approaches to the Quadratic Minimum Spanning Tree Problem

Roberto Cordone Gianluca Passeri

Dipartimento di Tecnologie dell'Informazione
Università degli Studi di Milano

Definition of the QMSTP

Let

- $G = (V, E)$ a *connected undirected graph* ($n = |V|$ and $m = |E|$)
- $c : E \rightarrow \mathbb{Z}$ a *linear cost* function
- $q : E \times E \rightarrow \mathbb{Z}$ a *quadratic cost* function ($q_{ee} = 0$ and $q_{ef} = q_{fe}$)

Find

- a *spanning tree* $T = (V, X)$

Minimize

- the *total cost* $z_X = \sum_{e \in X} c_e + \sum_{e,f \in X} q_{ef}$

An IQP formulation

Objective function

$$\min z(x) = \sum_{e \in E} c_e \cdot x_e + \sum_{e \in E} \sum_{f \in E} q_{ef} \cdot x_e \cdot x_f$$

Constraints

Acyclicity : $\sum_{e \in E(S)} x_e \leq |S| - 1 \qquad S \subseteq V, |S| \geq 2$

Cardinality : $\sum_{e \in E} x_e = n - 1$

Integrality : $x_e \in \{0, 1\} \qquad e \in E$

An IQP formulation

Objective function

$$\min z(x) = \sum_{e \in E} c_e \cdot x_e + \sum_{e \in E} \sum_{f \in E} q_{ef} \cdot x_e \cdot x_f$$

Constraints

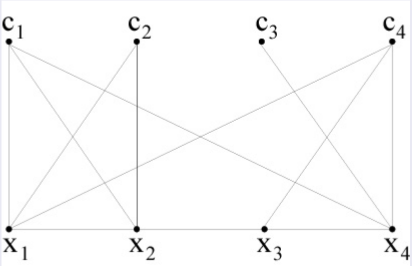
$$\begin{array}{lll} \text{Acyclicity}: & \displaystyle\sum_{e \in E(S)} x_e \leq |S| - 1 & S \subseteq V, |S| \geq 2 \\ \text{Cardinality}: & \displaystyle\sum_{e \in E} x_e = n - 1 & \\ \text{Integrality}: & x_e \in \{0, 1\} & e \in E \end{array}$$

Computational complexity

Strongly $\mathcal{NP}$-complete: reduction from *SAT*

1. A vertex x_i for each variable, a vertex c_l for each clause
2. An edge for each occurrence (x_i, c_l) of variable x_i in clause c_l
3. An edge for each pair (x_i, x_{i+1}) with $i = 1 \dots n-1$

$$f = (x_1 + x_2 + x_4) \cdot (\overline{x}_1 + \overline{x}_2) \cdot (\overline{x}_4) \cdot (x_1 + x_3 + \overline{x}_4)$$

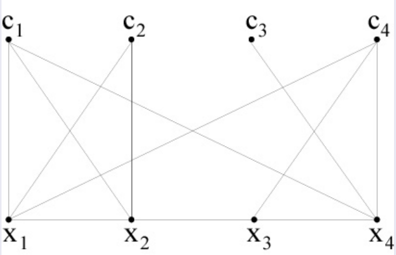


Computational complexity

Strongly $\mathcal{NP}$-complete: reduction from *SAT*

4. $c_e = 0$ for all $e \in E$
5. $q_{ef} = 1$ when e and f are opposite occurrences of the same variable
6. $q_{ef} = 0$ for all other pairs of edges

$$f = (x_1 + x_2 + x_4) \cdot (\overline{x}_1 + \overline{x}_2) \cdot (\overline{x}_4) \cdot (x_1 + x_3 + \overline{x}_4)$$



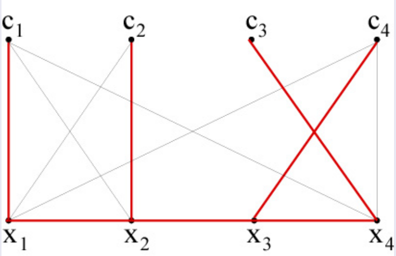
$q = 1$ for
(x_1, c_1, x_1, c_2)
(x_1, c_2, x_1, c_4)
(x_2, c_1, x_2, c_2)
(x_4, c_1, x_4, c_3)
(x_4, c_1, x_4, c_4)

Computational complexity

Strongly $\mathcal{NP}$-complete: reduction from *SAT*

- An edge identifies a satisfying occurrence
- A spanning tree identifies a satisfying assignment
- If $z_X = 0$, tree X employs only zero cost edges
- Zero cost edges identify reciprocally consistent occurrences

$$f = (x_1 + x_2 + x_4) \cdot (\overline{x}_1 + \overline{x}_2) \cdot (\overline{x}_4) \cdot (x_1 + x_3 + \overline{x}_4)$$



$q = 1$ for
(x_1, c_1, x_1, c_2)
(x_1, c_2, x_1, c_4)
(x_2, c_1, x_2, c_2)
(x_4, c_1, x_4, c_3)
(x_4, c_1, x_4, c_4)

Approximability and number of solutions

Approximability

- Since the optimal cost is $z^* = 0$,
  the QMSTP is non-approximable unless $\mathcal{P} = \mathcal{NP}$

Number of solutions

- It is exactly given by Kirchoff's theorem (1847)
- For complete graphs (Cayley's theorem)

$$|\mathcal{T}| = n^{n-2}$$

Approximability and number of solutions

Approximability

- Since the optimal cost is $z^* = 0$,
  the QMSTP is non-approximable unless $\mathcal{P} = \mathcal{NP}$

Number of solutions

- It is exactly given by Kirchoff's theorem (1847)
- For complete graphs (Cayley's theorem)

$$|\mathcal{T}| = n^{n-2}$$

Survey on the literature

Survey on the literature

- simple greedy heuristics (Xu, 1995)
- genetic algorithm (Zhou and Gen, 1998)
- two genetic algorithms for the fuzzy QMSTP (Gao and Lu, 2005)

Algorithmic approaches developed

1. ILP formulation (*standard linearization*)
2. Heuristic approaches
   a) Average contribution method (*constructive*)
   b) Minimum contribution method (*constructive*)
   c) Sequential fixing method (*constructive* and *adaptive*)
   d) *Tabu Search*
3. Exact approach
   - *Branch and Bound* (combinatorial bound)

Algorithmic approaches developed

1. ILP formulation (*standard linearization*)
2. Heuristic approaches
   a) Average contribution method (*constructive*)
   b) Minimum contribution method (*constructive*)
   c) Sequential fixing method (*constructive* and *adaptive*)
   d) *Tabu Search*
3. Exact approach
   - *Branch and Bound* (combinatorial bound)

Algorithmic approaches developed

1. ILP formulation (*standard linearization*)
2. Heuristic approaches
   a) Average contribution method (*constructive*)
   b) Minimum contribution method (*constructive*)
   c) Sequential fixing method (*constructive* and *adaptive*)
   d) *Tabu Search*
3. Exact approach
   - *Branch and Bound* (combinatorial bound)

Common elements for the constructive heuristics

Starting from the quadratic objective function...

$$\min z(x) = \sum_{e \in E} c_e \cdot x_e + \sum_{e \in E} \sum_{f \in E} q_{ef} \cdot x_e \cdot x_f$$

...approximate it with a linear one...

$$z(x) = \sum_{e \in E} x_e \cdot \left(c_e + \sum_{f \in E} q_{ef} \cdot x_f \right) \approx \sum_{e \in E} \tilde{c}_e \cdot x_e$$

$$\text{where} \qquad \tilde{c}_e \approx c_e + \sum_{f \in E} q_{ef} \cdot x_f \qquad e \in E$$

...and solve the resulting MSTP

$$z(x) = \min_{x \in \mathcal{T}} \sum_{e \in E} \tilde{c}_e \cdot x_e$$

Average Contribution Method

Objective function

$$z(x) = \sum_{e \in E} x_e \cdot \left(c_e + \sum_{f \in E} q_{ef} \cdot x_f \right)$$

Approximated objective function

$$\tilde{c}_e = c_e + (n-2) \cdot \frac{\sum_{f \in E} q_{ef}}{m-1} \qquad e \in E$$

Complexity

$$\Theta(m^2 + m \log n)$$

Minimum Contribution Method

Objective function

$$z(x) = \sum_{e \in E} x_e \cdot \left(c_e + \sum_{f \in E} q_{ef} \cdot x_f \right)$$

Approximated objective function

$$\tilde{c}_e = c_e + \sum_{f \in E_e^*} q_{ef} \qquad e \in E$$

E_e^* includes the $n-2$ edges with minimum q_{ef}

Complexity

$$\Theta(m^2 \log n + m \log n)$$

Sequential Fixing Method

Objective function

$$z(x) = \sum_{e \in E} x_e \cdot \left(c_e + \sum_{f \in E} q_{ef} \cdot x_f \right)$$

Approximated (adaptive) objective function

Set $X := \emptyset$ and $F := E$

Move from F to X the edge with minimum

$$\tilde{c}_e = c_e + \sum_{f \in X} q_{ef} + (n - 2 - |X|) \cdot \frac{\sum_{f \in F \setminus \{e\}} q_{ef}}{|F| - 1} \quad e \in F$$

Remove from F the edges which close loops with X

Complexity

$\Theta(m^2 n + mn\alpha(m, n))$

Local search

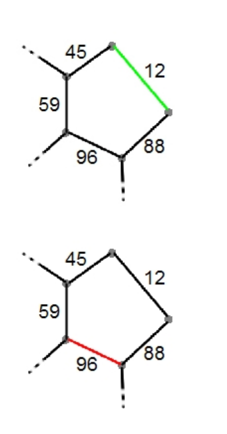
Main elements

- Neighbourhood:
  one edge in / one edge out
- Feasible edges out (for each edge in): loop formed by the edge in
- Evaluation in $\Theta(1)$ time of the objective function variation

Complexity of a move

$$O((m-n) \cdot n)$$

Move



Local search

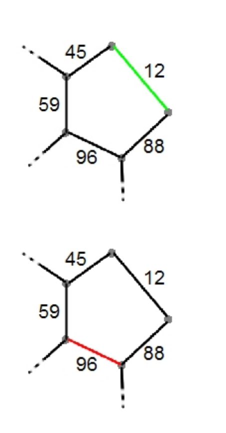
Main elements

- Neighbourhood:
  one edge in / one edge out
- Feasible edges out (for each edge in):
  loop formed by the edge in
- Evaluation in $\Theta(1)$ time of the objective function variation

Complexity of a move

$$O((m-n)\cdot n)$$

Move



Local search

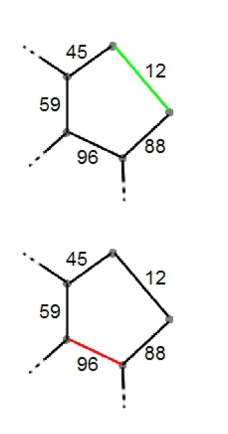
Main elements

- Neighbourhood:
  one edge in / one edge out
- Feasible edges out (for each edge in):
  loop formed by the edge in
- Evaluation in $\Theta(1)$ time of the objective function variation

Complexity of a move

$$O((m-n) \cdot n)$$

Move



Local search

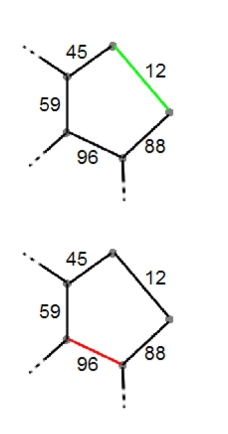
Main elements

- Neighbourhood:
  one edge in / one edge out
- Feasible edges out (for each edge in):
  loop formed by the edge in
- Evaluation in $\Theta(1)$ time of the objective function variation

Complexity of a move

$$O((m-n)\cdot n)$$

Move



Evaluation of a move

Auxiliary data structure

- The contribution of each edge $e \in E$ to the total cost is

$$D_e = c_e + \sum_{f \in X} q_{ef}$$

as

$$z_X = \sum_{e \in X} D_e$$

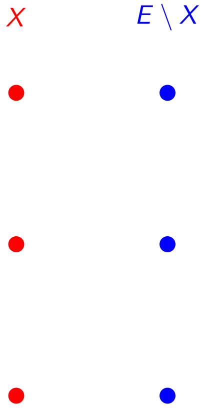
- Replace e by f: the total cost z

1. decreases by D_e
2. increases by D_f
3. decreases by q_{ef}

$$z_{X_{(\mathrm{new})}} = z_{X_{(\mathrm{old})}} - D_e + D_f - q_{ef}$$

- Finally, update D_i for all $i \in E$

$$D_i := D_i - q_{ie} + q_{if}$$



Evaluation of a move

Auxiliary data structure

- The contribution of each edge $e \in E$ to the total cost is

$$D_e = c_e + \sum_{f \in X} q_{ef}$$

as

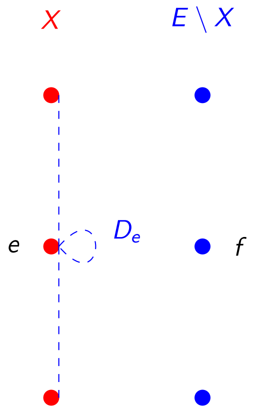
$$z_X = \sum_{e \in X} D_e$$

- Replace e by f: the total cost z
    1. decreases by D_e
    2. increases by D_f
    3. decreases by q_{ef}

$$z_{X_{(\mathrm{new})}} = z_{X_{(\mathrm{old})}} - D_e + D_f - q_{ef}$$

- Finally, update D_i for all $i \in E$

$$D_i := D_i - q_{ie} + q_{if}$$



Evaluation of a move

Auxiliary data structure

- The contribution of each edge $e \in E$ to the total cost is

$$D_e = c_e + \sum_{f \in X} q_{ef}$$

as

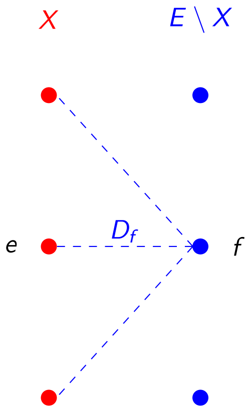
$$z_X = \sum_{e \in X} D_e$$

- Replace e by f: the total cost z
  1. decreases by D_e
  2. increases by D_f
  3. decreases by q_{ef}

$$z_{X_{(\text{new})}} = z_{X_{(\text{old})}} - D_e + D_f - q_{ef}$$

- Finally, update D_i for all $i \in E$

$$D_i := D_i - q_{ie} + q_{if}$$



Evaluation of a move

Auxiliary data structure

- The contribution of each edge $e \in E$ to the total cost is

$$D_e = c_e + \sum_{f \in X} q_{ef}$$

as

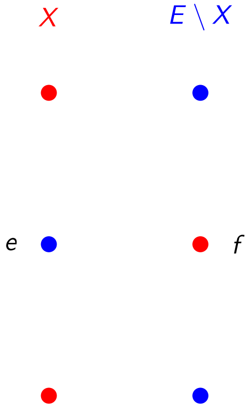
$$z_X = \sum_{e \in X} D_e$$

- Replace e by f: the total cost z
  1. decreases by D_e
  2. increases by D_f
  3. decreases by q_{ef}

$$z_{X_{(\mathrm{new})}} = z_{X_{(\mathrm{old})}} - D_e + D_f - q_{ef}$$

- Finally, update D_i for all $i \in E$

$$D_i := D_i - q_{ie} + q_{if}$$



Evaluation of a move

Auxiliary data structure

- The contribution of each edge $e \in E$ to the total cost is

$$D_e = c_e + \sum_{f \in X} q_{ef}$$

as

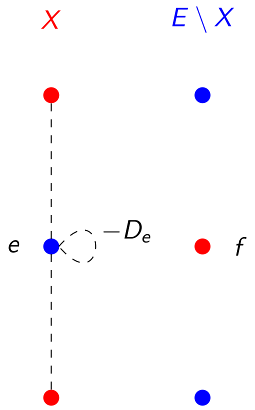
$$z_X = \sum_{e \in X} D_e$$

- Replace e by f: the total cost z
  1. decreases by D_e
  2. increases by D_f
  3. decreases by q_{ef}

$$z_{X_{(\mathrm{new})}} = z_{X_{(\mathrm{old})}} - D_e + D_f - q_{ef}$$

- Finally, update D_i for all $i \in E$

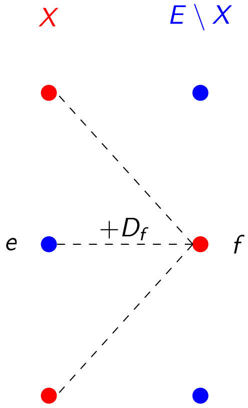
$$D_i := D_i - q_{ie} + q_{if}$$



Evaluation of a move

Auxiliary data structure

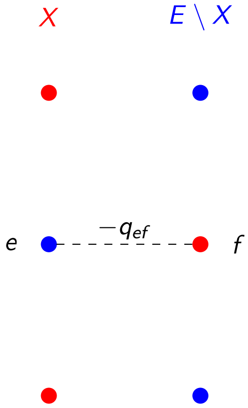
- The contribution of each edge $e \in E$ to the total cost is
$$D_e = c_e + \sum_{f \in X} q_{ef}$$
as
$$z_X = \sum_{e \in X} D_e$$
- Replace e by f: the total cost z
  1. decreases by D_e
  2. increases by D_f
  3. decreases by q_{ef}
$$z_{X_{(\text{new})}} = z_{X_{(\text{old})}} - D_e + D_f - q_{ef}$$
- Finally, update D_i for all $i \in E$
$$D_i := D_i - q_{ie} + q_{if}$$



Evaluation of a move

Auxiliary data structure

- The contribution of each edge $e \in E$ to the total cost is
$$D_e = c_e + \sum_{f \in X} q_{ef}$$
as
$$z_X = \sum_{e \in X} D_e$$
- Replace e by f: the total cost z
  1. decreases by D_e
  2. increases by D_f
  3. decreases by q_{ef}
$$z_{X_{(\text{new})}} = z_{X_{(\text{old})}} - D_e + D_f - q_{ef}$$
- Finally, update D_i for all $i \in E$
$$D_i := D_i - q_{ie} + q_{if}$$



Evaluation of a move

Auxiliary data structure

- The contribution of each edge $e \in E$ to the total cost is

$$D_e = c_e + \sum_{f \in X} q_{ef}$$

  as

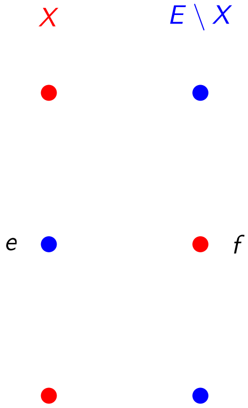
$$z_X = \sum_{e \in X} D_e$$

- Replace e by f: the total cost z
  1. decreases by D_e
  2. increases by D_f
  3. decreases by q_{ef}

$$z_{X_{(\mathrm{new})}} = z_{X_{(\mathrm{old})}} - D_e + D_f - q_{ef}$$

- Finally, update D_i for all $i \in E$

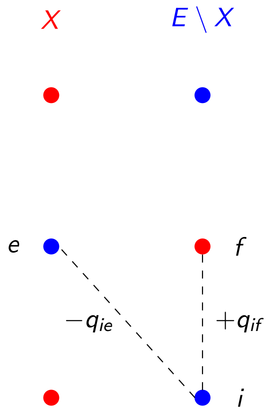
$$D_i := D_i - q_{ie} + q_{if}$$



Evaluation of a move

Auxiliary data structure

- The contribution of each edge $e \in E$ to the total cost is
$$D_e = c_e + \sum_{f \in X} q_{ef}$$
as
$$z_X = \sum_{e \in X} D_e$$
- Replace e by f: the total cost z
  1. decreases by D_e
  2. increases by D_f
  3. decreases by q_{ef}
$$z_{X_{(\mathrm{new})}} = z_{X_{(\mathrm{old})}} - D_e + D_f - q_{ef}$$
- Finally, update D_i for all $i \in E$
$$D_i := D_i - q_{ie} + q_{if}$$



Tabu Search

Main elements

- Tabu attribute: last iteration in/out for each edge
- Two independent tabu lists: longer tabu for insertion than for deletion ($E \setminus X \gg X$)
- Adaptive tabu tenure:
  - increasing when solution worsens
  - decreasing when solution improves
- Stop: maximum number of iterations

Tabu Search

Main elements

- Tabu attribute: last iteration in/out for each edge
- Two independent tabu lists: longer tabu for insertion than for deletion ($E \setminus X \gg X$)
- Adaptive tabu tenure:
  - increasing when solution worsens
  - decreasing when solution improves
- Stop: maximum number of iterations

Tabu Search

Main elements

- Tabu attribute: last iteration in/out for each edge
- Two independent tabu lists: longer tabu for insertion than for deletion ($E \setminus X \gg X$)
- Adaptive tabu tenure:
  - increasing when solution worsens
  - decreasing when solution improves
- Stop: maximum number of iterations

Main elements

- Lower bound: linear approximation from below $\rightarrow$ Kruskal's algorithm
- Upper bound: Kruskal's solution provides one for free
- Branching edge:
  cheapest unfixed edge in the relaxed solution
- Visit strategy: Best-Lower-Bound first
  (or hybrid: Best-Upper-Bound first followed by Best-Lower-Bound first)

Main elements

- Lower bound: linear approximation from below $\rightarrow$ Kruskal's algorithm
- Upper bound: Kruskal's solution provides one for free
- Branching edge:
  cheapest unfixed edge in the relaxed solution
- Visit strategy: Best-Lower-Bound first
  (or hybrid: Best-Upper-Bound first followed by Best-Lower-Bound first)

Main elements

- Lower bound: linear approximation from below $\longrightarrow$ Kruskal's algorithm
- Upper bound: Kruskal's solution provides one for free
- Branching edge:
  cheapest unfixed edge in the relaxed solution
- Visit strategy: Best-Lower-Bound first
  (or hybrid: Best-Upper-Bound first followed by Best-Lower-Bound first)

Main elements

- Lower bound: linear approximation from below $\rightarrow$ Kruskal's algorithm
- Upper bound: Kruskal's solution provides one for free
- Branching edge:
  cheapest unfixed edge in the relaxed solution
- Visit strategy: Best-Lower-Bound first
  (or hybrid: Best-Upper-Bound first followed by Best-Lower-Bound first)

Lower bounds implemented

Relaxation

$$z(x) = \sum_{e \in E} c_e \; x_e + \sum_{e \in E} \sum_{f \in E} q_{ef} \; x_e \; x_f \geq z_1(x), z_2(x), z_3(x), \;\; \forall x \in \mathcal{T}$$

Three lower bounds

$$z_1(x) = \sum_{e \in F} c_e \cdot x_e$$

$$z_2(x) = \sum_{e \in X} c_e + \sum_{e \in X} \sum_{f \in X} q_{ef} + \sum_{e \in F} x_e \cdot \left(c_e + \sum_{e \in X} q_{ef} \right)$$

$$z_3(x) = \sum_{e \in X} c_e + \sum_{e \in X} \sum_{f \in X} q_{ef} + \sum_{e \in F} x_e \cdot \left(c_e + \sum_{e \in X} q_{ef} \right) + \\ + \sum_{e \in F} x_e \cdot \left(\sum_{f \in X_e^*} q_{ef} \right)$$

Lower bounds implemented

Relaxation

$$z(x) = \sum_{e \in E} c_e \ x_e + \sum_{e \in E} \sum_{f \in E} q_{ef} \ x_e \ x_f \geq z_1(x), z_2(x), z_3(x), \ \forall x \in \mathcal{T}$$

Three lower bounds

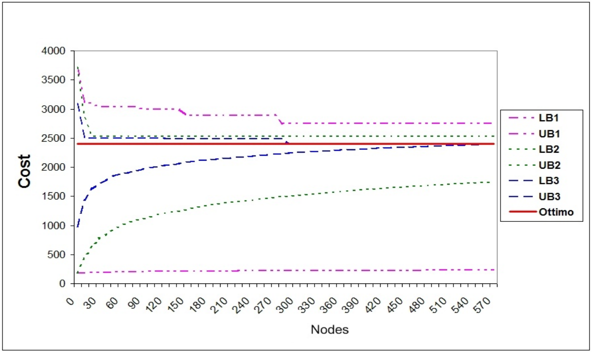
$$z_1(x) = \sum_{e \in F} c_e \cdot x_e$$

$$z_2(x) = \sum_{e \in X} c_e + \sum_{e \in X} \sum_{f \in X} q_{ef} + \sum_{e \in F} x_e \cdot \left(c_e + \sum_{e \in X} q_{ef} \right)$$

$$z_3(x) = \sum_{e \in X} c_e + \sum_{e \in X} \sum_{f \in X} q_{ef} + \sum_{e \in F} x_e \cdot \left(c_e + \sum_{e \in X} q_{ef} \right) + \\ + \sum_{e \in F} x_e \cdot \left(\sum_{f \in X_e^*} q_{ef} \right)$$

Comparison between three lower bounds

Comparison between the three lower bounds.
The graph considered has $n = 10$, 67% density and $c, q \in [1; 100]$



Machine features

Machine features	
Processor	2 Dual Core AMD Opteron Processor 275 2.2GHz
RAM	3 Gb
HD	250 Gb
Operating system	Linux
Language	ANSI-C
Compiler	gcc

Instance features

Graph properties

- Number of vertices n
- Density $\rho = 2m/n(n-1)$
- Linear costs c uniformly random
- Quadratic q uniformly random

Values

- 5 classes: 10, 15, 20, 25, 30
- 3 classes: 33%, 67%, 100%
- 2 classes: 1-10, 1-100
- 2 classes: 1-10, 1-100

Total number of instances

$$5 \cdot 3 \cdot 2 \cdot 2 = 60$$

Average gap of the heuristic algorithms

Average gap wrt the best known result

Vertici	ACM	MCM	SFM	TS(SFM)
10	24,70%	27,50%	3,68%	0,00%
15	21,20%	28,74%	4,66%	0,00%
20	22,14%	28,74%	5,47%	0,05%
25	23,31%	30,40%	4,42%	0,15%
30	26,16%	31,21%	4,09%	0,21%

Computational time: $<$ 1 sec for the constructive algorithms,
$<$ 1 min for Tabu Search (100000 iterations)

ILP solver and Branch and Bound compared

Average gap between upper and lower bound after 2 hours (branch-and-bound initialized by Tabu Search, and this by SFM)

	Density					
	33%		67%		100%	
n	**Solver**	**B&B**	**Solver**	**B&B**	**Solver**	**B&B**
10	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
15	0.00%	0.00%	56.60%	10.39%	95.93%	20.40%
20	59.12%	16.98%	92.58%	37.98%	98.40%	41.54%
25	-	34.30%	-	54.07%	-	57.15%
30	-	46.37%	-	60.67%	-	69.11%

Conclusions and future developments

Conclusions

- ILP formulation impractical
- Constructive algorithms fast, but quite ineffective
- Local search fast and effective (not always optimal)
- Branch and bound algorithm: viable for small instances ($n \leq 15$)

Future developments

- Tighter lower bounds (combinatorial, Lagrangean, semidefinite)
- Quadratic Minimum Spanning Arborescence (QMSAP)

Conclusions and future developments

Conclusions

- ILP formulation impractical
- Constructive algorithms fast, but quite ineffective
- Local search fast and effective (not always optimal)
- Branch and bound algorithm: viable for small instances ($n \leq 15$)

Future developments

- Tighter lower bounds (combinatorial, Lagrangean, semidefinite)
- Quadratic Minimum Spanning Arborescence (QMSAP)

Conclusions and future developments

Conclusions

- ILP formulation impractical
- Constructive algorithms fast, but quite ineffective
- Local search fast and effective (not always optimal)
- Branch and bound algorithm: viable for small instances ($n \leq 15$)

Future developments

- Tighter lower bounds (combinatorial, Lagrangean, semidefinite)
- Quadratic Minimum Spanning Arborescence (QMSAP)

Conclusions and future developments

Conclusions

- ILP formulation impractical
- Constructive algorithms fast, but quite ineffective
- Local search fast and effective (not always optimal)
- Branch and bound algorithm: viable for small instances ($n \leq 15$)

Future developments

- Tighter lower bounds (combinatorial, Lagrangean, semidefinite)
- Quadratic Minimum Spanning Arborescence (QMSAP)

Conclusions and future developments

Conclusions

- ILP formulation impractical
- Constructive algorithms fast, but quite ineffective
- Local search fast and effective (not always optimal)
- Branch and bound algorithm: viable for small instances ($n \leq 15$)

Future developments

- Tighter lower bounds (combinatorial, Lagrangean, semidefinite)
- Quadratic Minimum Spanning Arborescence (QMSAP)

Conclusions and future developments

Conclusions

- ILP formulation impractical
- Constructive algorithms fast, but quite ineffective
- Local search fast and effective (not always optimal)
- Branch and bound algorithm: viable for small instances ($n \leq 15$)

Future developments

- Tighter lower bounds (combinatorial, Lagrangean, semidefinite)
- Quadratic Minimum Spanning Arborescence (QMSAP)