Instituto de computação

Power Optimization in Ad Hoc Wireless Networks with Biconnectivity Requirements

Celso C. Ribeiro
Renato N. Moraes

Universidade Federal Fluminense, Brazil

December 2009

Contents

1. Wireless ad hoc networks
2. Problem statement
3. System model and related work: problem variants
4. Integer programming formulation
   - Variables
   - Model
5. GRASP heuristic
   - Construction phase
   - Local search phase
   - Path-relinking
6. Numerical results
   - Experimental settings
   - Optimal solutions
   - GRASP approximate solutions
7. Concluding remarks

Contents

1. Wireless ad hoc networks
2. Problem statement
3. System model and related work: problem variants
4. Integer programming formulation
   - Variables
   - Model
5. GRASP heuristic
   - Construction phase
   - Local search phase
   - Path-relinking
6. Numerical results
   - Experimental settings
   - Optimal solutions
   - GRASP approximate solutions
7. Concluding remarks

Wireless ad hoc networks

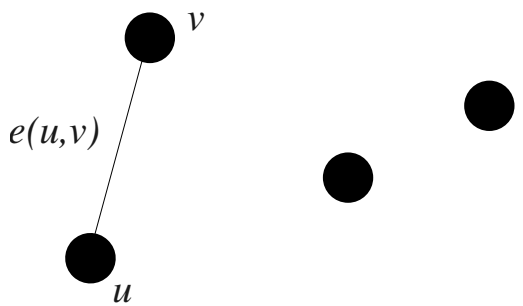
Definition

An *ad hoc network* consists of a collection of transceivers, in which a packet may have to traverse multiple consecutive wireless links to reach its final destination.

- Usefull when no communication infrastructure is available.
- Radio transceivers communicate without any fixed infrastructure.
- Examples of applications:
  - Battlefield communication
  - Disaster relief communication
  - Ubiquitous internet access
  - Sensor networks

Wireless ad hoc networks

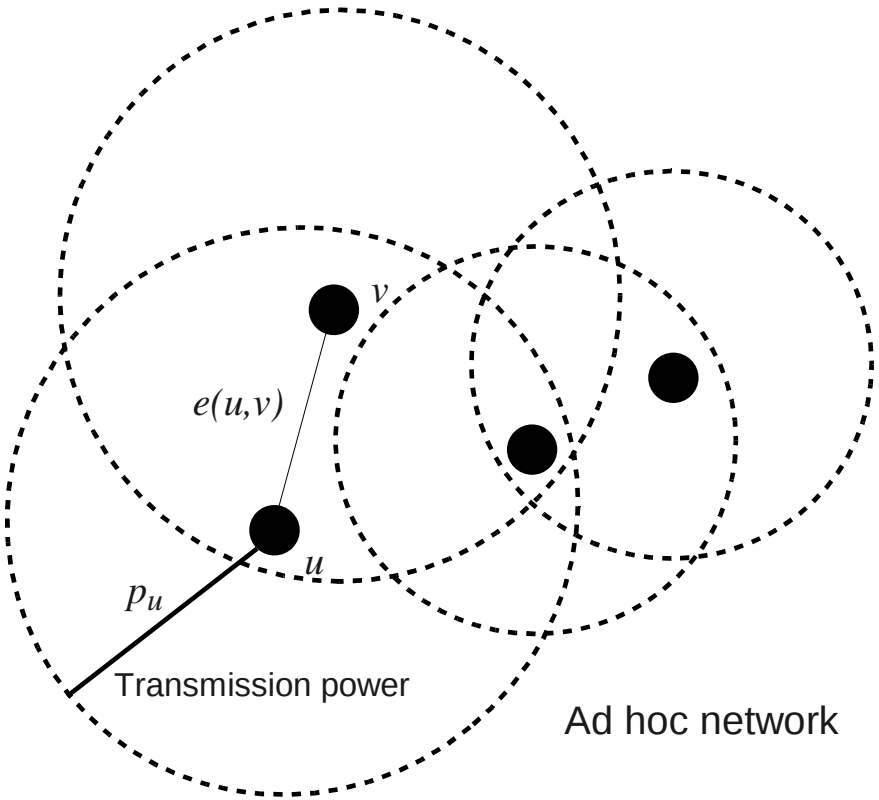
- Ad hoc networks can be represented by a set V of transceivers (nodes) together with their locations or the distances between them.
- For each ordered pair (u, v) of transceivers, with $u, v \in V$, we are given a non-negative arc weight $e(u, v) = d_{uv}^{\varepsilon}$
  - d_{uv} is the Euclidean distance between the transmitter u and the receiver v.
  - ε is the loss exponent, typically equal to two.



Ad hoc network

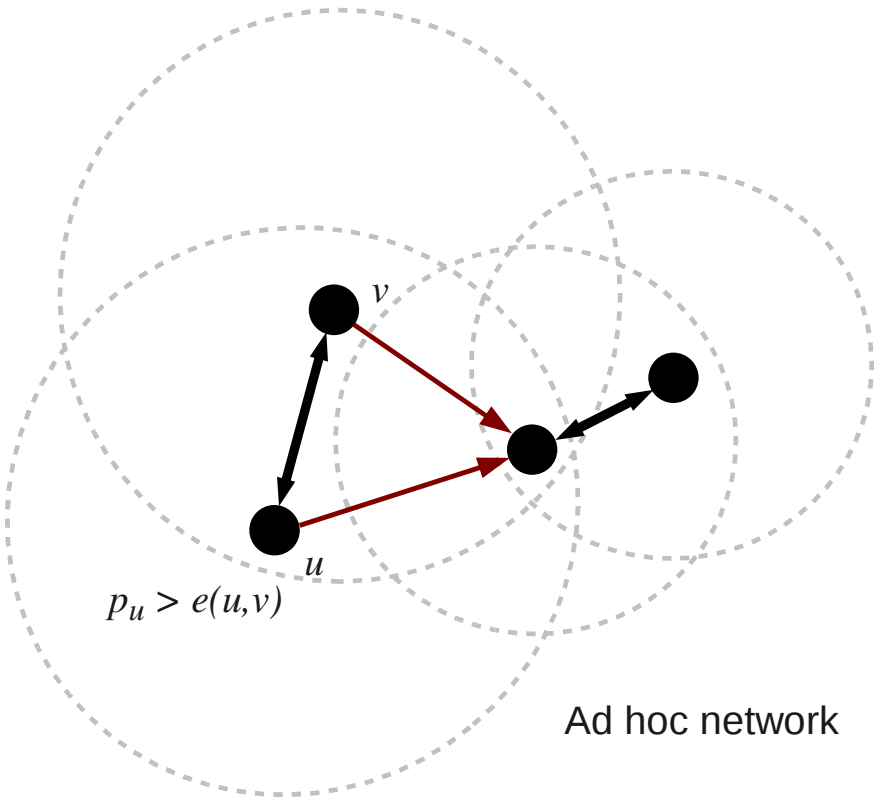
Wireless ad hoc networks

- A transmission power p_u is associated with each node $u \in V$.



Wireless ad hoc networks

- A signal transmitted by the transceiver u can be received at node v if and only if the transmission power assigned to node u is at least equal to $e(u, v)$, i.e. if $p_u \geq e(u, v)$.



Wireless ad hoc networks

- Wireless networks face a variety of constraints that do not appear in wired networks.

- Power constraints:
  - Nodes are battery powered: it is expensive and sometimes even infeasible to recharge the device.
  - Radios tend to be the major source of power dissipation.
  - Instead of transmitting with maximum power, algorithms adjust the transmission power of each node.

- Connectivity constraints:
  - Fault-tolerance requirements, due to their critical application domains and to the large number of failures.
  - If there is only one path between a pair of nodes, failure of a node or link between them will result in a disconnected graph.
  - Topologies with multiple, disjoint paths between any pair of nodes are often required.

Wireless ad hoc networks

- Wireless networks face a variety of constraints that do not appear in wired networks.
- Power constraints:
  - Nodes are battery powered: it is expensive and sometimes even infeasible to recharge the device.
  - Radios tend to be the major source of power dissipation.
  - Instead of transmitting with maximum power, algorithms adjust the transmission power of each node.
- Connectivity constraints:
  - Fault-tolerance requirements, due to their critical application domains and to the large number of failures.
  - If there is only one path between a pair of nodes, failure of a node or link between them will result in a disconnected graph.
  - Topologies with multiple, disjoint paths between any pair of nodes are often required.

Wireless ad hoc networks

- Wireless networks face a variety of constraints that do not appear in wired networks.
- Power constraints:
  - Nodes are battery powered: it is expensive and sometimes even infeasible to recharge the device.
  - Radios tend to be the major source of power dissipation.
  - Instead of transmitting with maximum power, algorithms adjust the transmission power of each node.
- Connectivity constraints:
  - Fault-tolerance requirements, due to their critical application domains and to the large number of failures.
  - If there is only one path between a pair of nodes, failure of a node or link between them will result in a disconnected graph.
  - Topologies with multiple, disjoint paths between any pair of nodes are often required.

Contents

1. Wireless ad hoc networks
2. Problem statement
3. System model and related work: problem variants
4. Integer programming formulation
   - Variables
   - Model
5. GRASP heuristic
   - Construction phase
   - Local search phase
   - Path-relinking
6. Numerical results
   - Experimental settings
   - Optimal solutions
   - GRASP approximate solutions
7. Concluding remarks

Problem statement

- Given the node set V and non-negative arc weights $e(u, v)$ for any $u, v \in V$:

Definition

The *biconnected minimum power consumption problem* consists of finding an optimal assignment of transmission powers $p : V \rightarrow R+$ to every node $u \in V$, such that the total power consumption $\sum_{u \in V} p_u$ is minimized and the resulting transmission graph $G = (V, E)$ is biconnected, where $E = \{(u, v) : u \in V, v \in V, p_u \geq e(u, v)\}$.

- Problem is proved to be NP-hard.

Contents

1 Wireless ad hoc networks

2 Problem statement

3 System model and related work: problem variants

4 Integer programming formulation
- Variables
- Model

5 GRASP heuristic
- Construction phase
- Local search phase
- Path-relinking

6 Numerical results
- Experimental settings
- Optimal solutions
- GRASP approximate solutions

7 Concluding remarks

System model

- Set V of transceivers, with $|V| = n$, each of them equipped with an omnidirectional antenna which is responsible for sending and receiving signals.
- Ad hoc network established by assigning a transmission power p_u to each transceiver $u \in V$.
- Each node can (possibly dynamically) adjust its transmitting power, based on the distance to the receiving nodes and on the background noise.

System model

- Power requirement at node u for supporting transmissions through a link from u to v:

$$e(u, v) \geq d_{uv}^{\varepsilon}.q_v,$$

where:

  - d_{uv}: Euclidean distance between the transmitter u and the receiver v,
  - ε: loss exponent
  - q_v: receiver's power threshold for signal detection, usually normalized to one

System model: symmetric input variant

- Power requirements for supporting transmissions between nodes u and v are often symmetric:

$$e(u, v) = e(v, u)$$

- Symmetric version widely accepted as reasonable:
  - Holds for free-space environments with non-obstructed lines of sight.
  - Disregards reflections, scattering, and diffraction caused e.g. by buildings and terrains.

System model: asymmetric input variant

- There may exist pairs of transceivers $u, v \in V$ such that

$$e(u, v) \neq e(v, u)$$

- Some situations:
  - Batteries with different power levels.
  - Heterogeneous nodes.
  - Different levels of ambient noise in the regions of each node.

System model: unidirectional topology

- Communication from u to v enabled whenever $p_u \geq e(u, v)$.
- Transmission graph $G = (V, E)$ associated with a power assignment $p : V \rightarrow R+$:

$$E = \{(u, v) : u \in V, v \in V, p_u \geq e(u, v)\}$$

- All arcs established by the power settings are considered to enforce biconnectedness.

System model: bidirectional topology

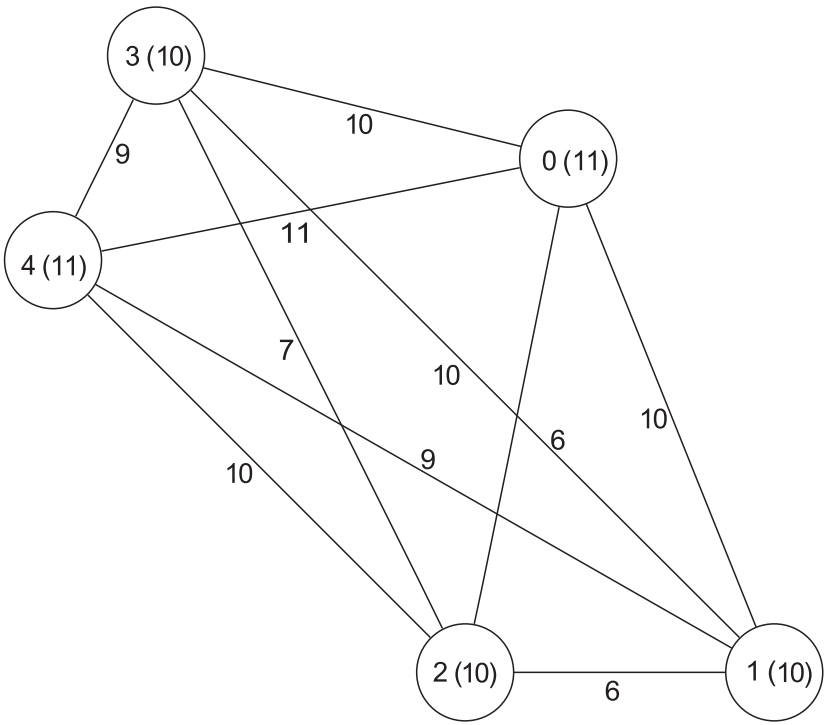
- Communication from u to v enabled whenever $p_u \geq e(u, v)$ and $p_v \geq e(v, u)$.
- Restricted arc set considered to enforce the biconnectivity constraints.
- Transmission graph $G(p) = (V, B(p))$ associated with a power assignment $p : V \rightarrow R+$:

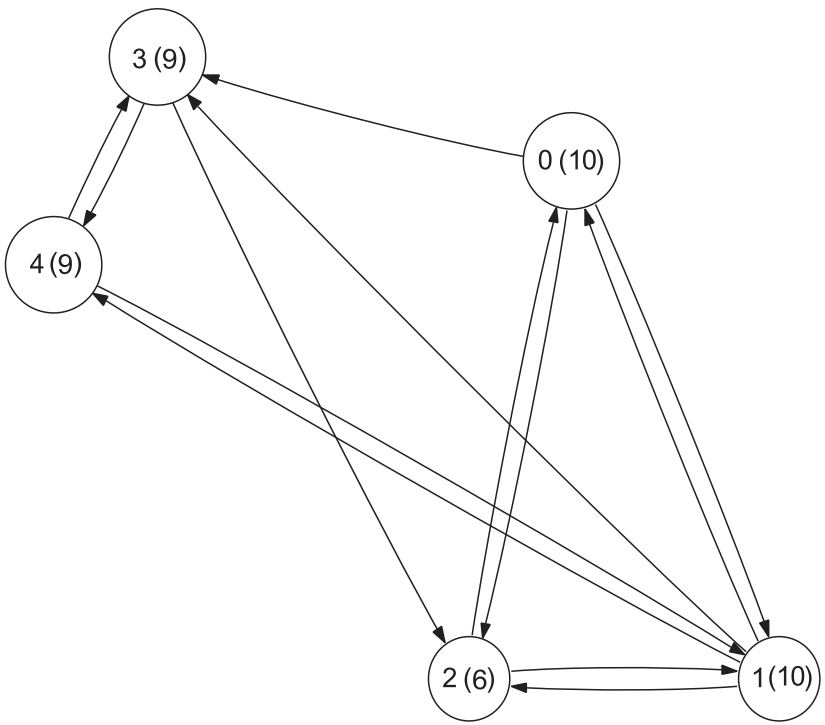
  $$B(p) = \{(u, v) : u \in V, v \in V, p_u \geq e(u, v), p_v \geq e(v, u)\} \subseteq E$$

- Edge $[u, v]$ is used as a communication link to enforce biconnectedness if v is within the transmission range of u and u is within the transmission range of v.

System model: example with symmetric input

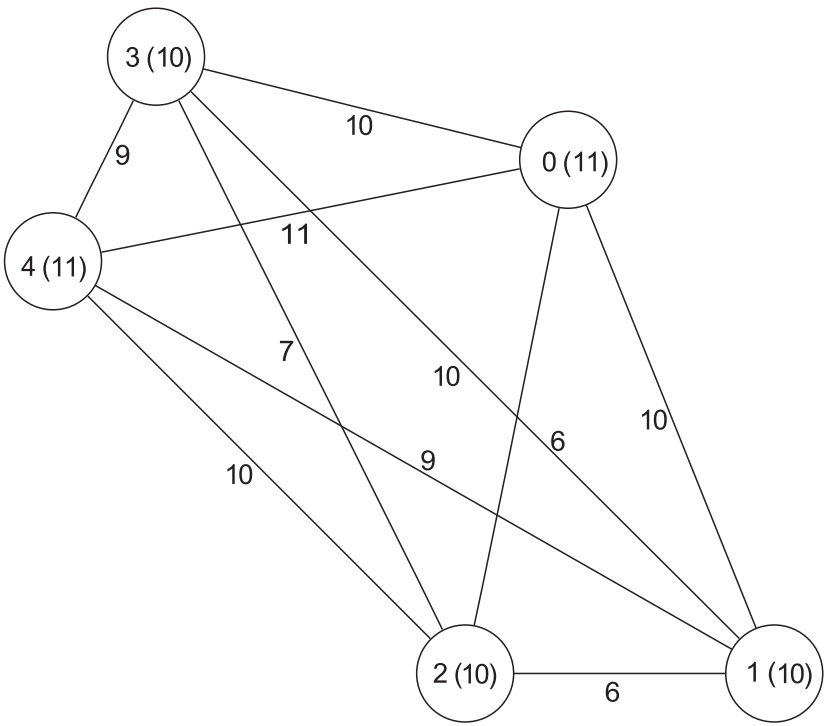
- Minimum biconnected unidirectional topology solution
- $G(p) = (V, E(p))$
- Total power consumption: 44

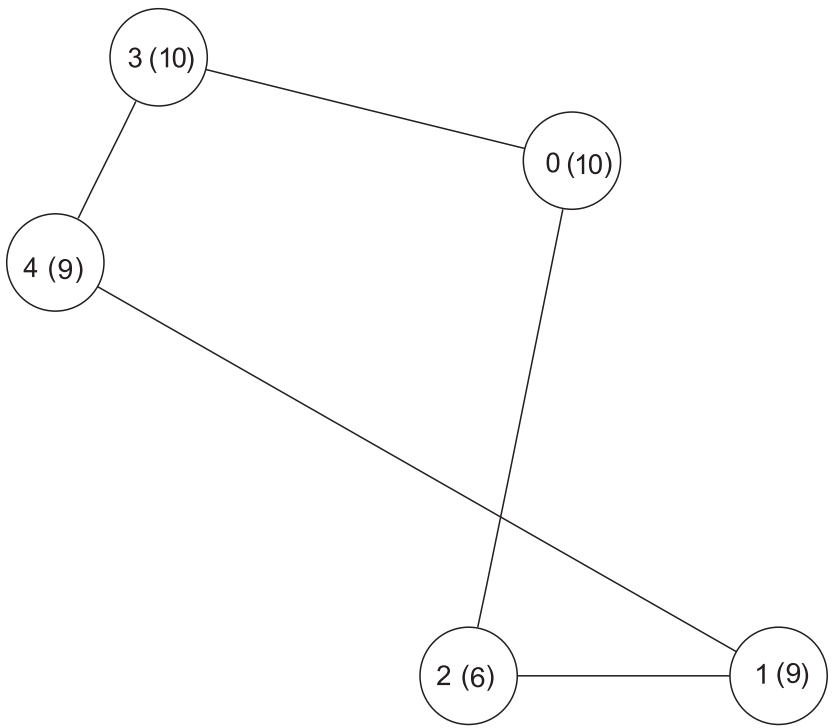




System model: example with symmetric input

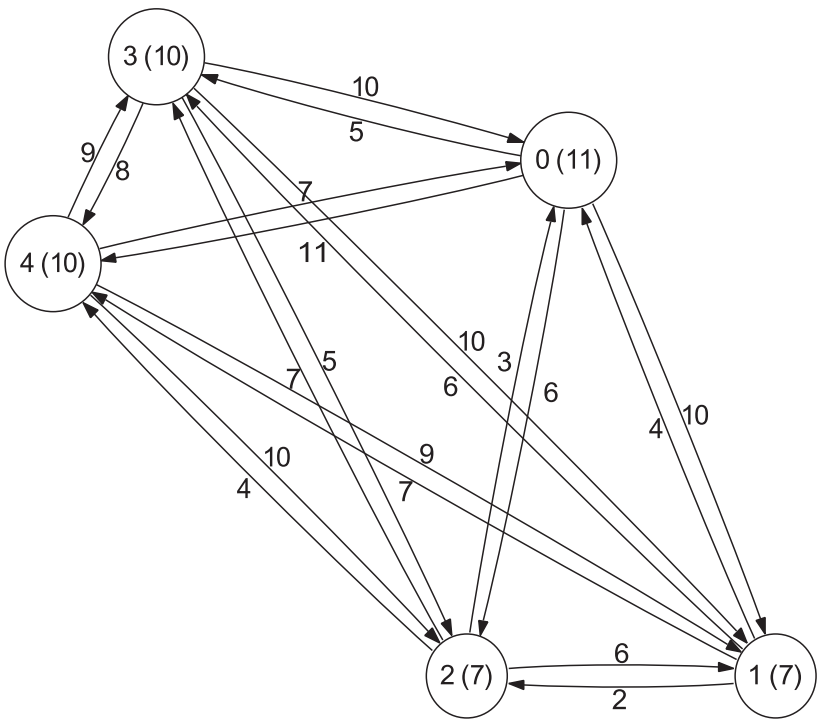
- Minimum biconnected bidirectional topology solution
- $G(p) = (V, B(p))$
- Total power consumption: 45

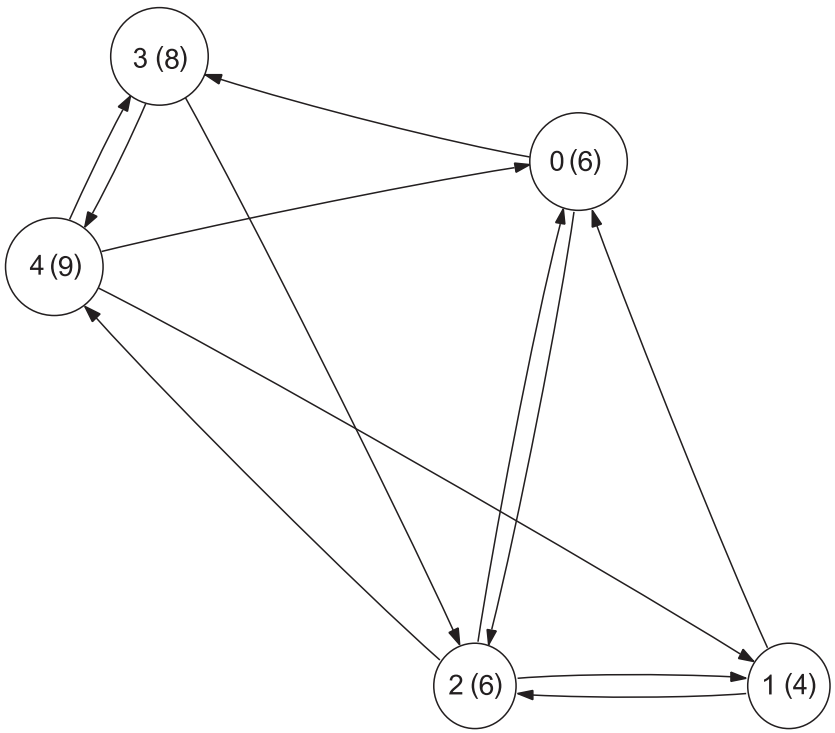




System model: example with asymmetric input

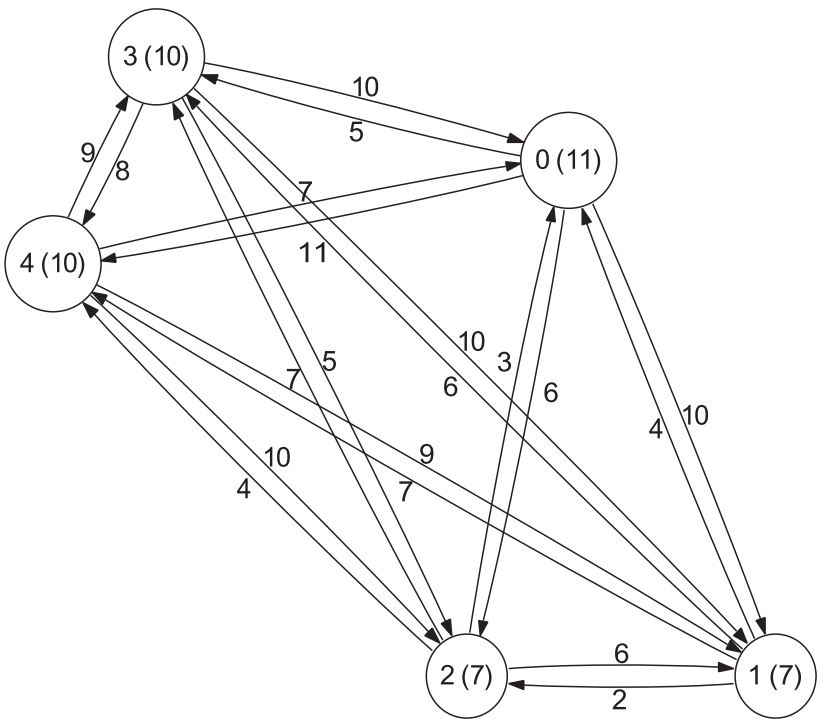
- Minimum biconnected unidirectional topology solution
- $G(p) = (V, E(p))$
- Total power consumption: 33

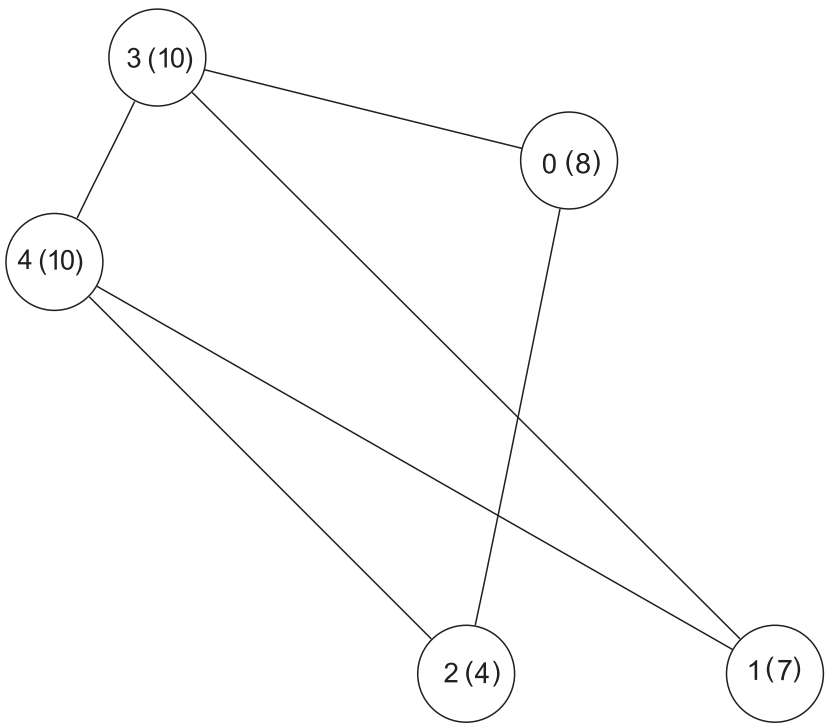




System model: example with asymmetric input

- Minimum biconnected bidirectional topology solution
- $G(p) = (V, B(p))$
- Total power consumption: 39





Problem variants

- Four versions of the biconnected minimum power consumption problem:
  - Symmetric input with unidirectional topology
  - Symmetric input with bidirectional topology
  - Asymmetric input with unidirectional topology
  - Asymmetric input with bidirectional topology

Related work: symmetric input

Unidirectional topology

- Connected transmission graph
  - Chen and Huang, 1989:
    - NP-hardness
    - 2-approximation algorithm
  - Kirousis et al., 2000:
    - NP-hardness in three-dimensional Euclidean space
    - 2-approximation algorithm
- Biconnected transmission graph
  - Calinescu and Wan, 2006:
    - NP-hardness
    - 4-approximation algorithm

Related work: symmetric input

Bidirectional topology

- Connected transmission graph
  - Calinescu etal., 2002:
    - NP-completeness
  - Althaus et al., 2006:
    - $(5/3 + \epsilon)$-approximation algorithm
    - Branch-and-cut algorithm
- Biconnected transmission graph
  - Lloyd et al., 2005:
    - $(2(2 - 2/n)(2 + 1/n))$-approximation algorithm

Related work: asymmetric input

Unidirectional topology

- Connected transmission graph
  - Krumke et al., 2003, Calinescu et al., 2003, Caragiannis et al., 2006:
    - $O(\log n)$-approximation algorithm

Bidirectional topology

- Connected transmission graph
  - Caragiannis et al., 2006:
    - $O(1.35 \log n)$-approximation algorithm
  - Calinescu et al., 2003:
    - $O(\log n)$-approximation algorithm

Contents

1 Wireless ad hoc networks

2 Problem statement

3 System model and related work: problem variants

4 Integer programming formulation
- Variables
- Model

5 GRASP heuristic
- Construction phase
- Local search phase
- Path-relinking

6 Numerical results
- Experimental settings
- Optimal solutions
- GRASP approximate solutions

7 Concluding remarks

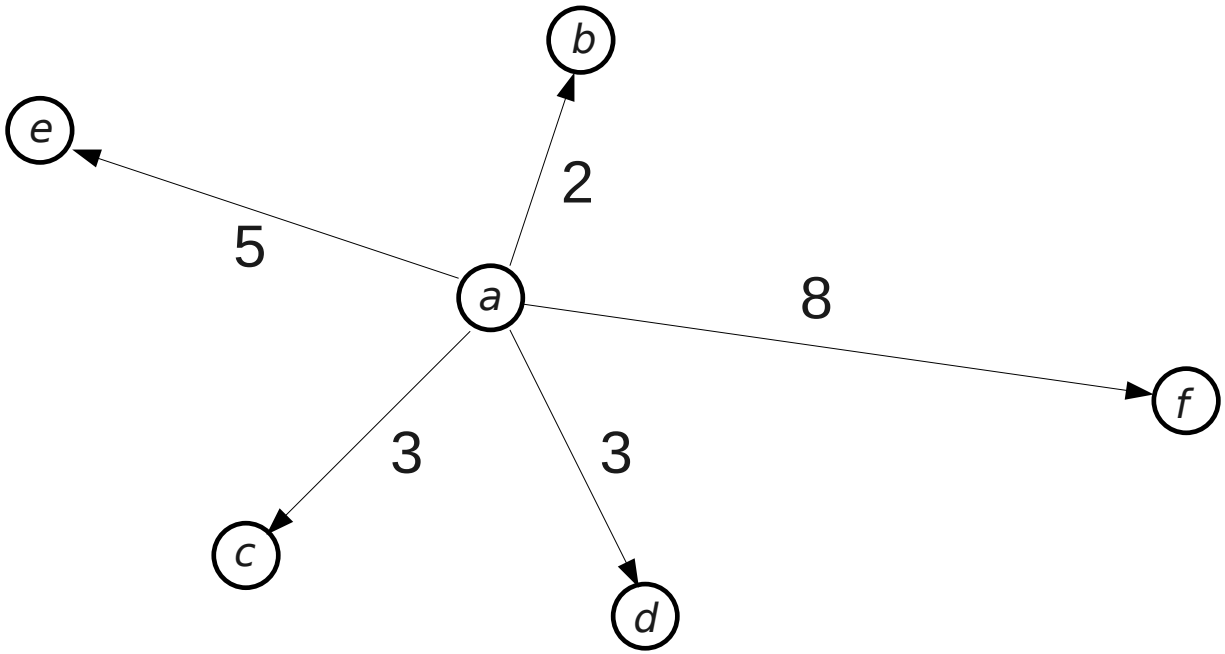
Integer programming formulation: variables

- Multicommodity network flow model (Magnanti and Raghavan, 2005)
- C is a set of $\lceil |V|/2 \rceil$ commodities.
- For each commodity $c \in C$:
  - Origin $o(c)$
  - Destination $d(c)$
- For any node $i \in V$ and any commodity $c \in C$:
  - $D_c(i) = -2$, if $i = o(c)$,
  - $D_c(i) = +2$, if $i = d(c)$,
  - $D_c(i) = 0$ otherwise.
- For node $i \in V$ and commodity $c \in C$, the binary variable f_{ij}^c represents the flow of commodity c through arc (i, j):
  - $f_{ij}^c = 1$, if arc (i, j) is used by commodity c for communication from node i to j,
  - $f_{ij}^c = 0$, otherwise.

Integer programming formulation: variables

- $P_i = [p_i^1, \ldots, p_i^{\phi(i)}]$ is a list of increasing "efficient" power levels that can be assigned to node $i \in V$, where:
  - p_i^1 is the minimum power such that transmissions from i reach at least one node in $V \setminus \{i\}$.
  - $p_i^{\ell+1} > p_i^\ell > 0$ for $\ell = 1, \ldots, \phi(i) - 1$.
  - $p_i^0 = 0$ for ease of representation.
- $T_i^\ell \neq \emptyset$ is the set of new nodes reachable from i if its power level increases from $p_i^{\ell-1}$ to p_i^ℓ, for $\ell = 1, \ldots, \phi(i)$.

Integer programming formulation: variables



- $P_a = [2, 3, 5, 8]$
- $T_a^1 = \{b\},\ T_a^2 = \{c, d\},\ T_a^3 = \{e\},\ T_a^4 = \{f\}$

Integer programming formulation: variables

- Nodes $i \in V$ and power levels $P_i = [p_i^1, \ldots, p_i^{\phi(i)}]$
- Binary variables x_i^ℓ, $\ell = 1, \ldots, \phi(i)$, determine the power level assigned to each node $i \in V$:
  - $x_i^\ell = 1$, if there is a node $j \in T_i^\ell$ such that link (i, j) is used for communication from i to j,
  - $x_i^\ell = 0$, otherwise.
- $p_i^{\bar{\ell}(i)}$ is the minimum power level such that transmissions from i reach at least two nodes in $V \setminus \{i\}$.

Integer programming formulation: model

$$\min \sum_{i \in V} \sum_{\ell=1}^{\phi(i)} (p_i^\ell - p_i^{\ell-1}) \cdot x_i^\ell$$

subject to:

$$\sum_{j \in V} f_{ji}^c - \sum_{l \in V} f_{il}^c = D_c(i), \qquad \forall c \in C, \forall i \in V$$

$$\sum_{j \in V} f_{ij}^c \leq 1, \qquad \forall c \in C, \forall i \in V : i \neq o(c), i \neq d(c)$$

$$x_i^\ell \geq f_{ij}^c, \qquad \forall i \in V, \forall c \in C, \forall j \in T_i^\ell, \ell = 1, \ldots, \phi(i)$$

$$x_i^{\ell+1} \leq x_i^\ell, \qquad \forall i \in V, \ell = 1, \ldots, \phi(i) - 1$$

$$x_i^\ell = 1, \qquad \forall i \in V, \ell = 1, \ldots, \bar{\ell}(i)$$

$$f_{ij}^c \in \{0, 1\}, \qquad \forall i, j \in V, \forall c \in C$$

$$x_i^\ell \in \{0, 1\}, \qquad \forall i \in V, \ell = 1, \ldots, \phi(i)$$

Integer programming formulation: model

- Flow conservation equations:

$$\sum_{j \in V} f_{ji}^c - \sum_{l \in V} f_{il}^c = D_c(i), \qquad \forall c \in C, \forall i \in V$$

- Node-disjointness:

$$\sum_{j \in V} f_{ij}^c \leq 1, \qquad \forall c \in C, \forall i \in V : i \neq o(c), i \neq d(c)$$

- $x_i^\ell = 1$ if there is a node $j \in T_i^\ell$ such that arc (i, j) is used for communication from i to j by commodity c:

$$x_i^\ell \geq f_{ij}^c, \qquad \forall i \in V, \forall c \in C, \forall j \in T_i^\ell, \ell = 1, \ldots, \phi(i)$$

Integer programming formulation: model

- $x_i^{\ell+1} = 0$ if the previous level was not used:

$$x_i^{\ell+1} \leq x_i^{\ell}, \qquad \forall i \in V, \ell = 1, \ldots, \phi(i) - 1$$

- Minimum power level $\bar{\ell}(i)$ reaches at least the two closest nodes:

$$x_i^{\ell} = 1, \qquad \forall i \in V, \ell = 1, \ldots, \bar{\ell}(i)$$

- Integrality requirements:

$$f_{ij}^{c} \in \{0, 1\}, \qquad \forall i, j \in V, \forall c \in C$$

$$x_i^{\ell} \in \{0, 1\}, \qquad \forall i \in V, \ell = 1, \ldots, \phi(i)$$

Contents

1 Wireless ad hoc networks

2 Problem statement

3 System model and related work: problem variants

4 Integer programming formulation
- Variables
- Model

5 GRASP heuristic
- Construction phase
- Local search phase
- Path-relinking

6 Numerical results
- Experimental settings
- Optimal solutions
- GRASP approximate solutions

7 Concluding remarks

GRASP fundamentals

- A greedy randomized adaptive search procedure (GRASP) is a multistart process.
- Each of its iterations consists of two phases:
  1. Construction: a feasible solution is built
  2. Local search (or improvement): a local optimum in the neighborhood of the current solution is sought.
- Best overall solution is returned.
- GRASP heuristic for the asymmetric input with bidirectional topology variant of the biconnected minimum power consumption problem.

Algorithm 1 Pseudo-code of a GRASP heuristic for minimization problems

Require: *Max_Iterations*,*Seed*
Ensure: Best known solution x^*

```
1: f* ← ∞;
2: for iteration = 1, ..., Max_Iterations do
3:     x ← Greedy_Randomized_Construction(Seed);
4:     x ← Local_Search(x);
5:     if cost(x) < f* then
6:         x* ← x;
7:         f* ← cost(x);
8: return x*;
```

GRASP construction phase

- Randomized construction phase has two stages:
  1. Build a power assignment p such that $G(p) = (V, B(p))$ is a bidirectional connected graph
  2. Change the power assignment p such that $G(p) = (V, B(p))$ is a bidirectional biconnected graph
- Greedy function that guides construction is based on the *wireless multicast advantage property*: if p_u is the current power assigned to node u and there is a node v such that $e(u, v) > p_u$, then the additional power required to set up communication from u to v is $e(u, v) - p_u$.

Greedy function for any $u, v \in V$

$$g(u, v) = \max\{0, e(u, v) - p_u\} + \max\{0, e(v, u) - p_v\}$$

- If $g(u, v) = 0$, then bidirectional communication between nodes u and v is already set up.

GRASP construction phase: first stage

Algorithm 2 First stage of the randomized construction phase

Require: Node set V, initial node $r \in V$, parameter $\alpha \in [0, 1]$
Ensure: Power assignment p and a bidirectional connected graph $G(p) = (V, B(p))$

1: $p_u \leftarrow 0$ for all $u \in V$;
2: $V' \leftarrow \{r\}$;
3: **while** $V \setminus V' \neq \emptyset$ **do**
4: **for all** $u \in V \setminus V'$ **do**
5: $g(u) \leftarrow \min_{v \in V'}\{g(u, v)\}$;
6: $Parent(u) \leftarrow v \in V' : g(u) = g(u, v)$;
7: $\underline{g} \leftarrow \min_{u \in V \setminus V'}\{g(u)\}$;
8: $\overline{g} \leftarrow \max_{u \in V \setminus V'}\{g(u)\}$;
9: Randomly select node $u \in V \setminus V'$ such that $g(u) \leq \underline{g} + \alpha(\overline{g} - \underline{g})$;
10: $p_u \leftarrow \max\{p_u, e(u, Parent(u))\}$;
11: $p_{Parent(u)} \leftarrow \max\{p_{Parent(u)}, e(Parent(u), u)\}$;
12: $V' \leftarrow V' \cup \{u\}$;
13: **return** p;

GRASP construction phase: second stage

- Biconnected component of a graph: maximal subset of nodes such that there are two disjoint paths between any two of them.
- Articulation point: node belonging to two or more biconnected components.
- Tarjan's algorithm is used to compute the biconnected components and articulation points of the current solution.
- Pairs of biconnected components are linked one by one.
- Second stage stops when a biconnected graph is built.

GRASP construction phase: second stage

Algorithm 3 Second stage of the randomized construction phase

Require: Node set V, power assignment p, and parameter $\alpha \in [0, 1]$

Ensure: Power assignment p and a bidirectional biconnected graph $G(p) = (V, B(p))$

1: **while** G is not biconnected **do**
2: **for all** $u \in V$ that is not an articulation point **do**
3: $g'(u) = \min_{v \in V}\{g(u, v) : u \neq v,\ v$ is not an articulation point and does not belong to the same component as $u\}$;
4: $Parent(u) \leftarrow v \in V : g'(u) = g(u, v)$;
5: $\underline{g}' = \min_{u \in V}\{g'(u) : u$ is not an articulation point$\}$;
6: $\overline{g}' = \max_{u \in V}\{g'(u) : u$ is not an articulation point$\}$;
7: Randomly select node $u \in V$ that is not an articulation point such that $g'(u) \leq \underline{g}' + \alpha(\overline{g}' - \underline{g}')$;
8: $p_u \leftarrow \max\{p_u, e(u, Parent(u))\}$;
9: $p_{Parent(u)} \leftarrow \max\{p_v, e(Parent(u), u)\}$;
10: **return** p;

GRASP local search phase: operations

- $P_i = [p_i^1, \ldots, p_i^{\phi(i)}]$ is a list of "efficient" increasing power levels that can be assigned to node $i \in V$.
- Basic operations applied to each node $i \in V$ operating at the power level p_i^ℓ:
  - Decrease its current power assignment from p_i^ℓ to $p_i^{\ell'}$, where ℓ' is the highest level which supports a bidirectional edge: total power is decreased by $p_i^\ell - p_i^{\ell'}$.
  - Increase its current power assignment from p_i^ℓ to $p_i^{\ell+1}$: total power is increased by at least $p_i^{\ell+1} - p_i^\ell$.

GRASP local search phase: moves

- Local search explores the neighborhood of the current solution, attempting to reduce the total power consumption.
- A move starts by decreasing the power assignment of as many nodes as needed to break biconnectivity, followed by the increase of the power assignments of as many nodes as necessary to restablish biconnectivity.
  - Decrease operations are performed in non-increasing order of power decrease (i.e., start by largest power decrease)
  - Increase operations are performed in non-decreasing order of power increase (i.e., start by smallest power increase)
- First improving move is accepted and the search moves to the new solution.
- Procedure stops when no further improving moves exist.

GRASP local search phase: acceleration strategies

- Number of power increase operations investigated may be reduced to speedup the local search.
- Whenever biconnectivity is destroyed by a power decrease, the biconnected components are computed and two acceleration schemes are implemented:
  1. *Reduced scheme*: restricts power increases to pair of nodes belonging to the same biconnected components of the pair of nodes affected by the previous decrease.
  2. *Extended scheme*: considers power increases involving any pair of nodes from different biconnected components.

GRASP local search phase: acceleration strategies

- Three local search strategies are implemented, based on different acceleration schemes:
  1. *reduced local search* uses the reduced scheme;
  2. *extended local search* uses the extended scheme; and
  3. *mixed local search* first makes use of the reduced scheme until no further improving moves can be found, then continues using the extended scheme.

GRASP: path-relinking

- Path-relinking is an intensification strategy to explore trajectories connecting elite solutions obtained by heuristic methods such as GRASP.
- Path-relinking is usually carried out between two solutions: one is called the **initial solution**, while the other is the **guiding solution**.
- One or more paths in the solution space graph connecting these solutions are explored in the search for better solutions.

GRASP: path-relinking

Algorithm 4 Path-relinking

Require: Pair of solutions $p^{(1)}$ and $p^{(2)}$.
Ensure: Solution p^*.

1: Compute set of moves $\Delta(p^{(1)}, p^{(2)})$;
2: $f^* \leftarrow \min\{f(p^{(1)}), f(p^{(2)})\}$;
3: $p^* \leftarrow \text{argmin}(f(p^{(1)}), f(p^{(2)}))$;
4: $p \leftarrow p^{(1)}$;
5: **while** $\Delta(p^{(1)}, p^{(2)}) \neq \emptyset$ **do**
6: $\quad \eta^* \leftarrow \text{argmin}\{(f(p \bigoplus \eta) : \eta \in \Delta(p, p^{(2)}))\}$;
7: $\quad p \leftarrow p \bigoplus \eta^*$;
8: $\quad$ **if** $f(p) < f^*$ **then**
9: $\quad\quad f^* \leftarrow f(p)$;
10: $\quad\quad p^* \leftarrow p$;
11: **return** p^*;

GRASP: path-relinking

- GRASP with path-relinking makes use of an elite set to store a pool of diverse high-quality solutions found during the search.
- Path-relinking is applied as an intensification strategy at the end of each local search phase to a single pair of solutions:
  - The local optimum $p^{(1)}$ obtained after the local search phase.
  - A high-quality solution $p^{(2)}$ randomly selected from the pool.
- Path-relinking is applied using the best among $p^{(1)}$ and $p^{(2)}$ as the initial solution and the other as the guiding solution.
- Moves in $\Delta(p^{(1)}, p^{(2)})$ change the power level of any node $u \in V$ from $p_u^{(1)}$ to $p_u^{(2)}$.
- Moves are randomly selected from a candidate list with the most promising ones in the path being investigated.

GRASP-M with path-relinking

Algorithm 5 Pseudo-code of GRASP-M with path-relinking

Require: *Max_Iterations*,*Seed*
Ensure: Best known solution p^*

```
1:  f* ← ∞;
2:  Elite_Set ← ∅;
3:  for iteration = 1, ..., Max_Iterations do
4:      p ← Greedy_Randomized_Construction(Seed);
5:      repeat
6:          p ← Local_Search_Reduced(p);
7:          if no improvement then
8:              p ← Local_Search_Extended(p);
9:      until no improvement
10:     if Elite_Set ≠ ∅ then
11:         p ← Path_Relinking(p,Elite_Set);
12:     Update_EliteSet(p,Elite_Set);
13:     if cost(p) < f* then
14:         p* ← p;
15:         f* ← cost(p);
16: return p*;
```

Contents

1. Wireless ad hoc networks
2. Problem statement
3. System model and related work: problem variants
4. Integer programming formulation
   - Variables
   - Model
5. GRASP heuristic
   - Construction phase
   - Local search phase
   - Path-relinking
6. Numerical results
   - Experimental settings
   - Optimal solutions
   - GRASP approximate solutions
7. Concluding remarks

Experimental settings

- Computational experiments carried out on two classes of randomly generated asymmetric test problems with 10 to 800 nodes:
  - Euclidean instances:
    - Nodes uniformly distributed in the unit square grid.
    - Weights $e(u, v) = F \cdot d_{u,v}^2$, with $F \in [0.8, 1.2]$ being a random perturbation generated from a uniform distribution.
  - Random instances:
    - Weights $e(u, v)$ randomly generated in $(0, 1]$.
- 15 test instances for each problem size and class.
- Intel Core 2 Quad machine with a 2.40 GHz clock and 8 Gbytes of RAM memory running under GNU/Linux 2.6.24.
- CPLEX 11.0 used as the integer programming solver.
- GRASP heuristic implemented in C++ using GNU g++ version 4.1 as the compiler, with optimization parameter -O2.

Optimal solutions

- Average gaps between the linear and integer optimal values

Asymmetric							
Instance		unidirectional			bidirectional		
	$\|V\|$	solved	time (s)	gap (%)	solved	time (s)	gap (%)
Euclidean	10	15	0.89	11.06	15	0.47	7.51
	15	15	16.20	13.75	15	7.55	10.34
	20	15	177.59	13.40	15	66.61	8.10
	25	15	1563.94	11.96	15	298.53	7.71
	30	5	2837.09	7.47	12	1351.98	4.56
	50	–	–	–	–	–	–
Random	10	15	0.07	1.19	15	0.48	5.98
	15	15	0.16	0.00	15	6.99	10.83
	20	15	0.87	0.01	15	117.36	10.87
	25	15	2.36	0.01	15	872.44	13.48
	30	15	5.69	0.02	1	5559.86	13.55
	50	15	126.89	0.02	0	–	–

Symmetric							
Instance		unidirectional			bidirectional		
	$\|V\|$	solved	time (s)	gap (%)	solved	time (s)	gap (%)
Euclidean	10	15	0.78	10.90	15	0.48	7.25
	15	15	16.03	14.23	15	7.24	10.14
	20	15	179.02	12.80	15	47.26	8.27
	25	15	1600.28	12.15	15	509.83	7.70
	30	6	4875.97	11.51	12	1373.72	4.20
	50	–	–	–	–	–	–
Random	10	15	0.11	1.56	15	0.15	0.82
	15	15	0.74	0.40	15	0.23	0.22
	20	15	6.78	0.29	15	2.69	0.28
	25	15	20.43	0.32	15	10.95	0.12
	30	15	102.12	0.22	15	73.71	0.24
	50	12	2827.35	0.07	11	562.42	0.06

Optimal solutions

- Minimum power consumption problem is hard to solve.
- Computation times increase very fast with $|V|$.
- CPLEX could not solve to optimality in three hours of computation even moderately-sized networks with 30 nodes.
- Large integrality gaps.

GRASP variants

- GRASP variants, with respect to the local search phase strategy:
  - GRASP-R implements the reduced strategy.
  - GRASP-X implements the extended strategy.
  - GRASP-M implements the mixed strategy.

GRASP variants

- Average objective values over five runs (one instance) as the running time limit increases from five to 3125 seconds:

	Time (s)				
Algoritmo	5	25	125	625	3125
GRASP-R	1.28856	1.28774	1.28573	1.28512	1.28367
GRASP-X	1.29235	1.29040	1.28794	1.28624	1.28468
GRASP-M	1.28850	1.28744	1.28536	1.28459	1.28367

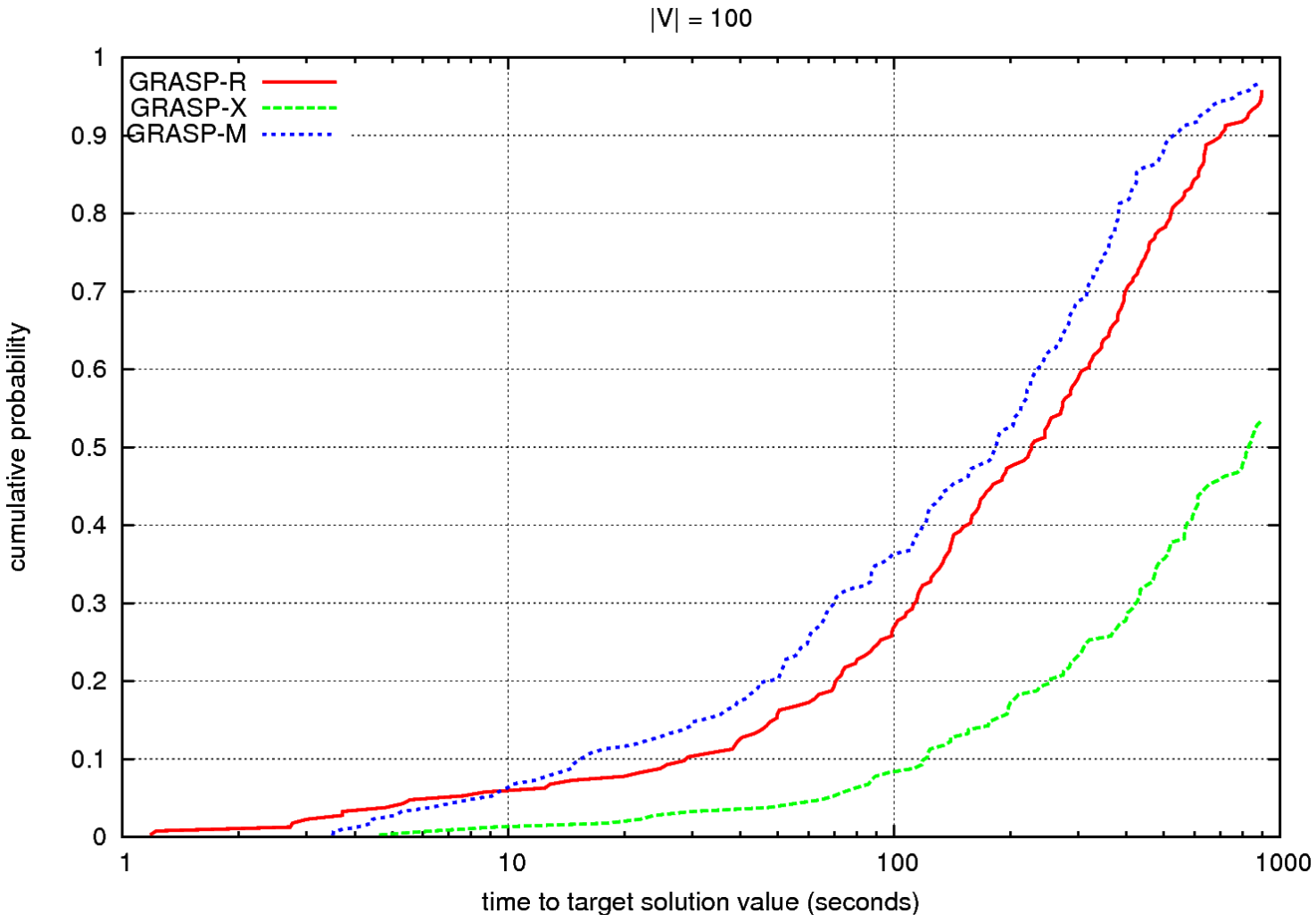
Table: $|V| = 100$

	Time (s)				
Algoritmo	5	25	125	625	3125
GRASP-R	1.73658	1.73364	1.72974	1.72841	1.72784
GRASP-X	1.73152	1.72941	1.72898	1.72813	1.72770
GRASP-M	1.73070	1.72892	1.72870	1.72816	1.72754

Table: $|V| = 200$

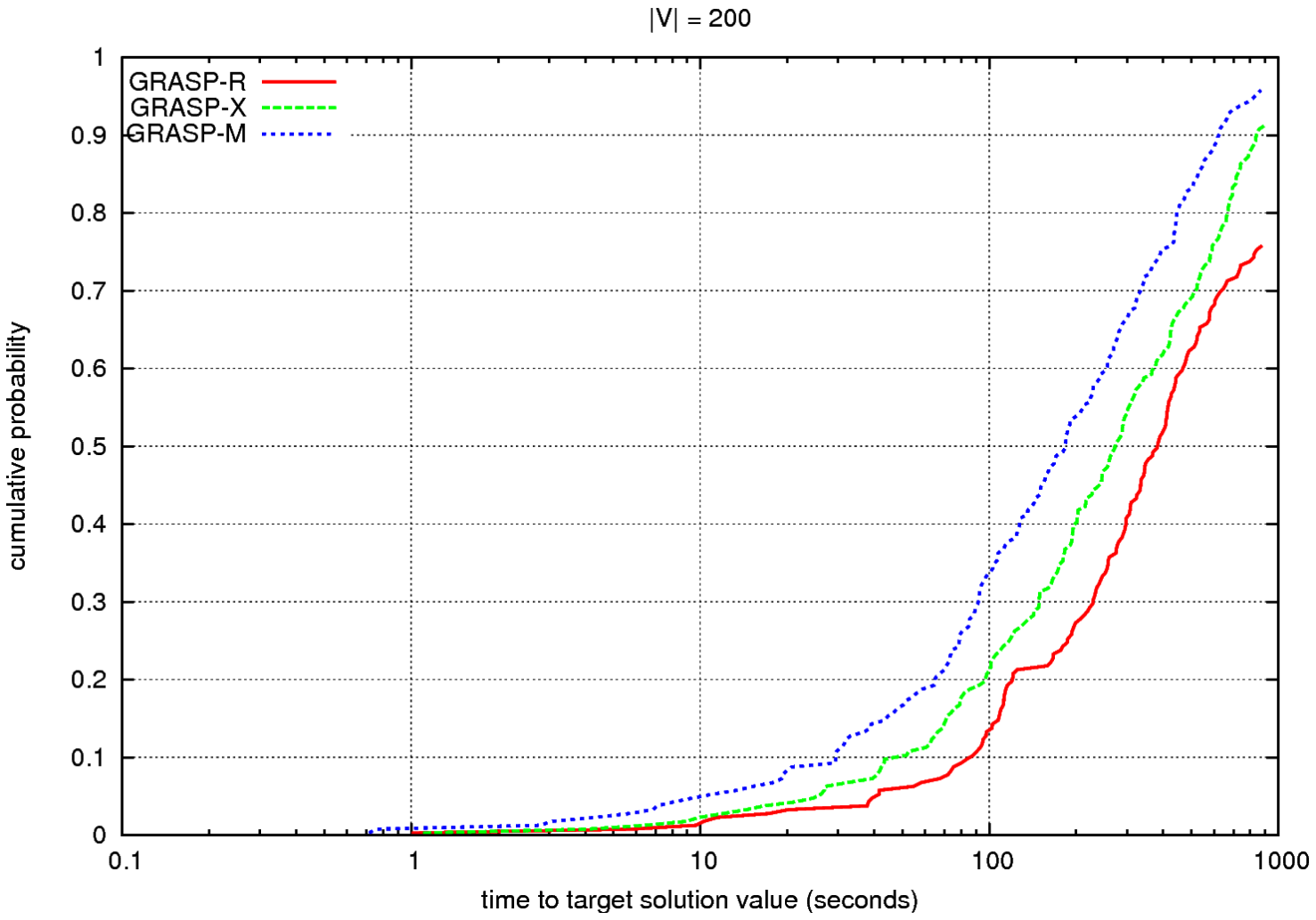
GRASP variants

- Run time distributions or time-to-target (ttt) plots
- Two hundred independent runs: stop when target is found.
- 15 minutes of computation time for each independent run.



GRASP variants

- Run time distributions or time-to-target (ttt) plots
- Two hundred independent runs: stop when target is found.
- 15 minutes of computation time for each independent run.



GRASP with path-relinking (GRASP-Mpr)

- Algorithms:
  - GRASP-M implements the mixed local search strategy
  - GRASP-Mpr implements the mixed local search strategy followed by path-relinking

GRASP with path-relinking (GRASP-Mpr)

- Average objective values over five runs (one instance) as the running time limit increases from five to 3125 seconds:

	Time (s)				
Algoritmo	5	25	125	625	3125
GRASP-M	1.28850	1.28744	1.28536	1.28459	1.28367
GRASP-Mpr	1.28946	1.28712	1.28528	1.28372	1.28303

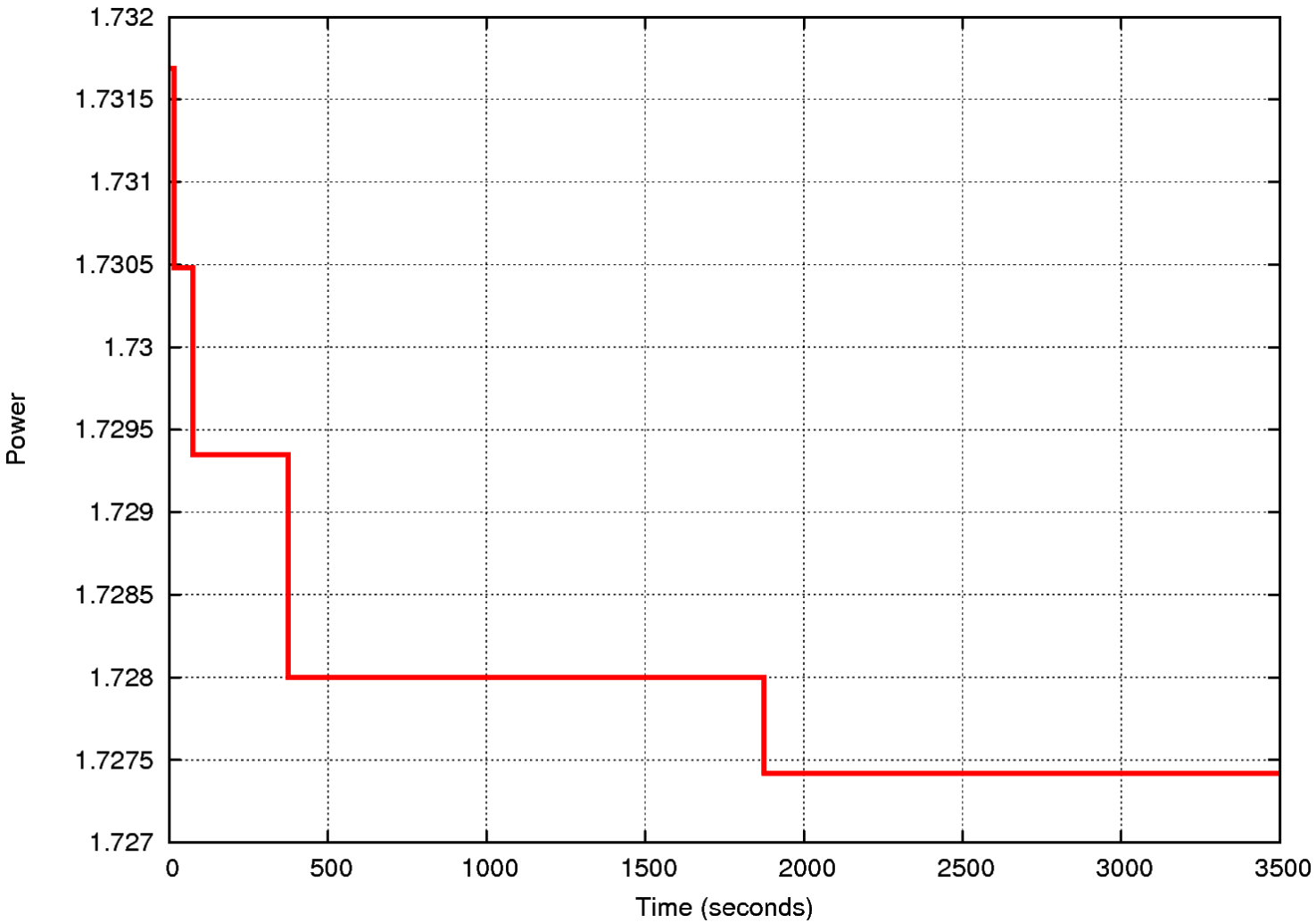
Table: $|V| = 100$

	Time (s)				
Algoritmo	5	25	125	625	3125
GRASP-M	1.73070	1.72892	1.72870	1.72816	1.72754
GRASP-Mpr	1.73169	1.73048	1.72935	1.72800	1.72742

Table: $|V| = 200$

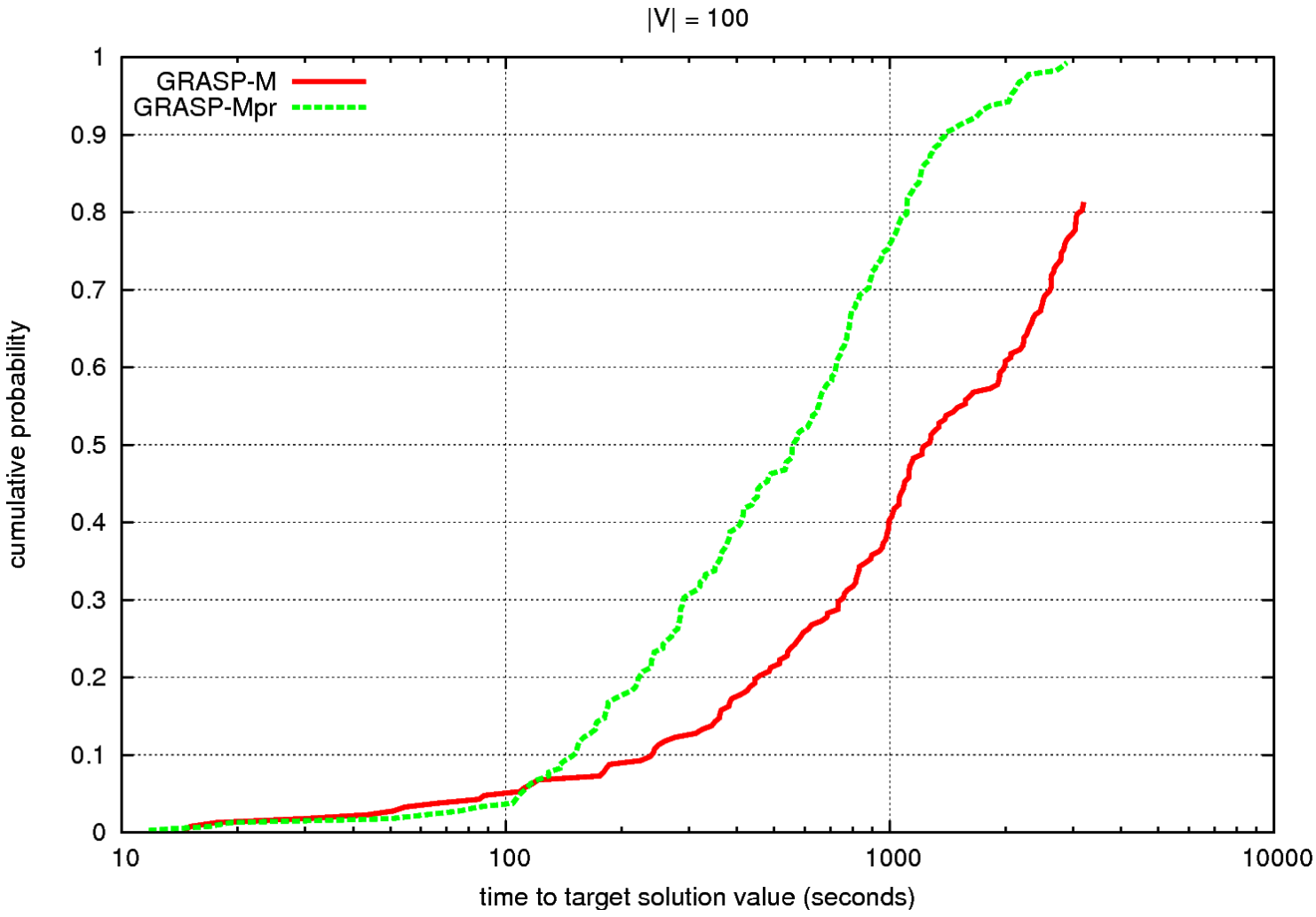
GRASP with path-relinking (GRASP-Mpr)

- Continuous improvement in solution quality along the computation time for one Euclidean instance with $|V| = 200$:



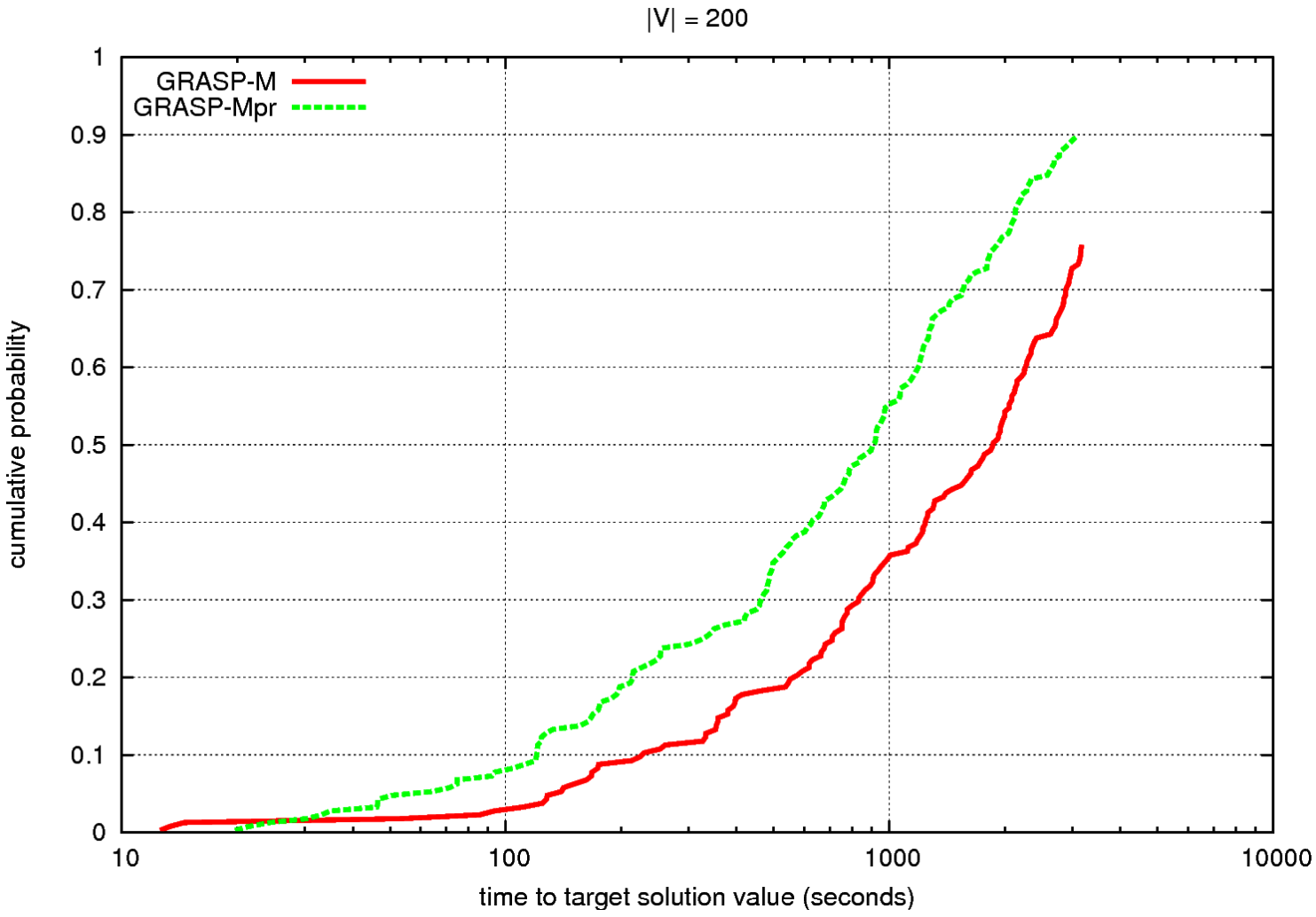
GRASP with path-relinking (GRASP-Mpr)

- Run time distributions or time-to-target (ttt) plots
- Two hundred independent runs: stop when target is found.
- One hour of computation time for each independent run.



GRASP with path-relinking (GRASP-Mpr)

- Run time distributions or time-to-target (ttt) plots
- Two hundred independent runs: stop when target is found.
- One hour of computation time for each independent run.



GRASP with path-relinking (GRASP-Mpr)

- GRASP-Mpr heuristic found the optimal solutions for all problems with up to 25 nodes.
- Computation times to find the known optimal solutions with $|V| = 25$:
  - Euclidean instances solved in less than one second.
  - Random instances are harder and took approximately five seconds on average.

	Exact		GRASP-Mpr	
Instances	Total power	Time (s)	Total power	Time (s)
Random	5.49494	3207.42	5.49494	5.72
Euclidean	5.44325	4332.45	5.44325	0.25

GRASP with path-relinking (GRASP-Mpr): comparisons

- GRASP-Mpr vs. MST-Augmentation of Calinescu and Wan, 2006:
  - GRASP-Mpr: 10 minutes of running time for each instance.
  - MST-Augmentation is faster: a few seconds for $|V| = 800$.
  - GRASP-Mpr finds much better solutions.

Instances		Total power			Degree (avg.)		
	$\|V\|$	GRASP-Mpr	MSTAug	red. (%)	GRASP-Mpr	MSTAug	red. (%)
Random	25	5.49494	13.05400	57.91	2.48	8.32	70.19
	50	8.41205	24.25722	65.32	2.43	13.24	81.65
	100	11.69800	44.67938	73.82	2.63	21.49	87.76
	200	17.21848	84.95794	79.73	2.67	37.57	92.89
	400	24.93783	158.78020	84.29	2.65	64.07	95.86
	800	37.30339	292.67002	87.25	2.70	107.00	97.48
Euclidean	25	5.44325	12.84646	21.32	2.55	4.20	39.29
	50	10.17110	24.20324	24.51	2.54	4.38	42.01
	100	19.34007	44.97218	27.27	2.53	4.61	45.12
	200	37.29308	85.08285	29.75	2.52	4.70	46.38
	400	73.72322	158.26410	28.36	2.55	4.61	44.69
	800	103.23984	293.63736	28.53	2.56	4.64	44.83

GRASP with path-relinking (GRASP-Mpr): comparisons

- GRASP-Mpr vs. greedy construction phase:
  - GRASP-Mpr: 10 minutes of running time for each instance.
  - Greedy is much faster: less than two seconds for $|V| = 800$.
  - GRASP-Mpr significantly improves all greedy solutions.
  - Average node degrees show that GRASP-Mpr solutions are very close to the theoretical lower bounds.

Instances		Total power			Degree (avg.)		
	$\|V\|$	GRASP-Mpr	Greedy	red. (%)	GRASP-Mpr	Greedy	red. (%)
Random	25	5.49494	6.24476	12.01	2.48	3.04	18.42
	50	8.41205	9.55131	11.93	2.43	3.05	20.33
	100	11.69800	13.21970	11.51	2.63	3.11	15.43
	200	17.21848	19.20336	10.34	2.67	3.22	17.08
	400	24.93783	27.60327	9.66	2.65	3.16	16.14
	800	37.30339	40.92659	8.85	2.70	3.20	15.63
Euclidean	25	5.44325	5.90808	7.87	2.55	3.06	16.67
	50	10.17110	10.95295	7.14	2.54	2.94	13.61
	100	19.34007	21.02384	8.01	2.53	3.00	15.67
	200	37.29308	40.35054	7.58	2.52	2.99	15.72
	400	73.72322	78.88391	6.54	2.55	2.96	13.85
	800	103.23984	109.68819	5.88	2.56	2.95	13.22

MST-Augmentation vs. greedy

- Greedy finds much better solutions than MST-Augmentation.
- Since MST-Augmentation was originaly designed for Euclidean instances, it is faster than greedy for them.

Instances		Total power		Time (seconds)	
	$\|V\|$	MST-Aug	Greedy	MST-Aug	Greedy
Random	25	13.05400	6.24476	0.00000	0.00040
	50	24.25722	9.55131	0.00120	0.00100
	100	44.67938	13.21970	0.00840	0.00280
	200	84.95794	19.20336	0.10361	0.01600
	400	158.78020	27.60327	1.26008	0.10201
	800	292.67002	40.92659	15.50057	0.79165
Euclidean	25	6.91850	5.90808	0.00000	0.00040
	50	13.47369	10.95295	0.00040	0.00280
	100	26.59133	21.02384	0.00150	0.00400
	200	53.08669	40.35054	0.00320	0.02320
	400	102.90816	78.88391	0.01240	0.19321
	800	144.44319	109.68819	0.04960	1.67651

Contents

1. Wireless ad hoc networks
2. Problem statement
3. System model and related work: problem variants
4. Integer programming formulation
   - Variables
   - Model
5. GRASP heuristic
   - Construction phase
   - Local search phase
   - Path-relinking
6. Numerical results
   - Experimental settings
   - Optimal solutions
   - GRASP approximate solutions
7. Concluding remarks

Concluding remarks

- Integer programming formulation for the bidirectional topology variant of the biconnected minimum power consumption problem.
- Formulation can be easily extended to problems with other connectivity requirements.
- Formulation applied to four variants of the problem:
  - Symmetric or asymmetric input graphs
  - Unidirectional or bidirectional solutions
- State-of-the-art integer programming solver could not solve large instances to optimality.

Concluding remarks

- GRASP-Mpr heuristic proposed to find good approximate solutions for real-size problems.
- Heuristic applied to large instances of the asymmetric input with bidirectional topology variant.
- Experimental results for large networks with up to 800 nodes showed that:
  - Greedy heuristic is very quick and effective for on-line applications.
  - GRASP-Mpr heuristic is relatively fast.
  - GRASP-Mpr heuristic found good solutions which significantly improved those obtained by a literature heuristic.
  - Solutions obtained by GRASP-Mpr heuristic are very close to the optimal solutions (average node degrees very close to the theoretical lower bounds).
  - Furthermore, solutions obtained by GRASP-Mpr heuristic have fewer arcs/edges and smaller power assignments, which are useful to mitigate interference.