Optimal Orientation On-line

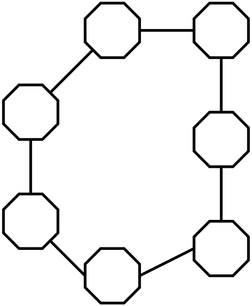
Lech Duraj Grzegorz Gutowski

Theoretical Computer Science Department
Jagiellonian University

SOFSEM 2008

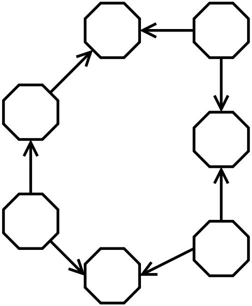
Building a one-way network

Imagine a network consisting of nodes and some links between them. These links mark pairs which can be connected.



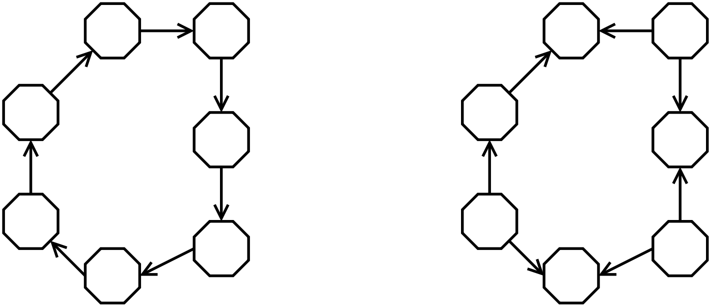
Building a one-way network

Imagine a network consisting of nodes and some links between them. These links mark pairs which can be connected. However, only one-way connections are available. We must build the best possible network, i.e. the one which allows the easiest communication.



Quality of solution

Some networks are clearly better then the others.



How to measure the quality of a network?

Quality measures

- *Reachable pairs problem*: maximize the number of pairs (u, v) s.t. v is reachable from u.

Quality measures

- *Reachable pairs problem*: maximize the number of pairs (u, v) s.t. v is reachable from u.
- *Average connectivity problem*: maximize the sum of $\lambda(u, v)$ (number of disjoint paths from u to v) over all pairs of vertices.

Off-line results

- For trees, reachable pairs and average connectivity are the same problem.

Off-line results

- For trees, reachable pairs and average connectivity are the same problem.
- There is a polynomial algorithm solving this case [Henning, Oellermann '04]

Off-line results

- For trees, reachable pairs and average connectivity are the same problem.
- There is a polynomial algorithm solving this case [Henning, Oellermann '04]
- The optimal solution gives $\Theta\left(n^2\right)$ connected pairs.

Off-line results

- For trees, reachable pairs and average connectivity are the same problem.
- There is a polynomial algorithm solving this case [Henning, Oellermann '04]
- The optimal solution gives $\Theta\left(n^2\right)$ connected pairs.
- For general graphs, reachable pairs problem can be solved using a similar algorithm, whereas average connectivity problem is NP-complete.

On-line game

Now, imagine a game between two players: Spoiler and Algorithm. The board is a growing graph G.

Spoiler

Algorithm

On-line game

Now, imagine a game between two players: Spoiler and Algorithm. The board is a growing graph G.

Spoiler

- adds a vertex with edges

Algorithm

On-line game

Now, imagine a game between two players: Spoiler and Algorithm. The board is a growing graph G.

Spoiler

- adds a vertex with edges

Algorithm

- directs new edges

On-line game

Now, imagine a game between two players: Spoiler and Algorithm. The board is a growing graph G.

Spoiler

- adds a vertex with edges

Algorithm

- directs new edges
- decisions are permanent

On-line game

Now, imagine a game between two players: Spoiler and Algorithm. The board is a growing graph G.

Spoiler

- adds a vertex with edges

Constraint: graph is connected

Algorithm

- directs new edges
- decisions are permanent

On-line game

Now, imagine a game between two players: Spoiler and Algorithm. The board is a growing graph G.

Spoiler

- adds a vertex with edges

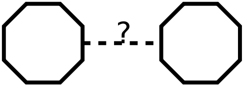
Constraint: graph is connected
Goal: minimize the number of connected pairs

Algorithm

- directs new edges
- decisions are permanent

Goal: maximize the number of connected pairs

Sample game



Spoiler

starts with a single edge

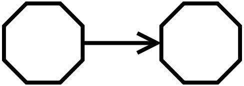
Optimal score

1

Algorithm score

$0+?$

Sample game



Algorithm

directs the edge

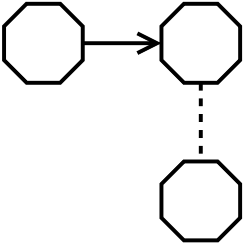
Optimal score

1

Algorithm score

1

Sample game



Spoiler

adds another edge

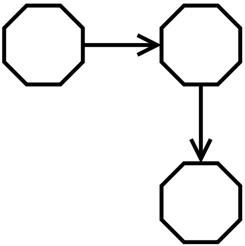
Optimal score

3

Algorithm score

$1+?$

Sample game



Algorithm

directs the edge

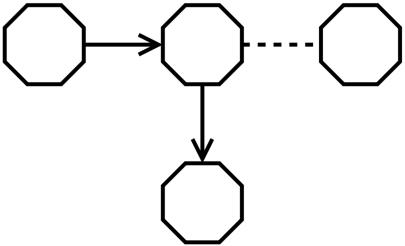
Optimal score

3

Algorithm score

3

Sample game



Spoiler

adds another edge

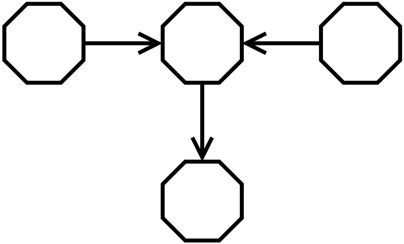
Optimal score

5

Algorithm score

3+?

Sample game



Algorithm

directs the edge

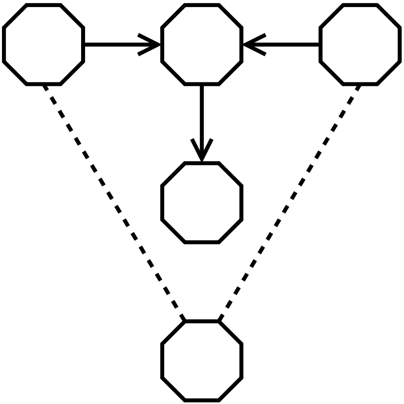
Optimal score

5

Algorithm score

5

Sample game



Spoiler

adds two edges

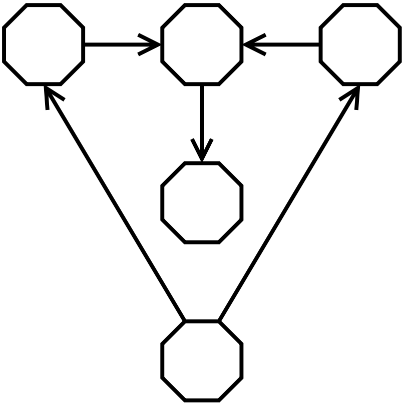
Optimal score

16

Algorithm score

$5+?$

Sample game



Algorithm

can't achieve optimum

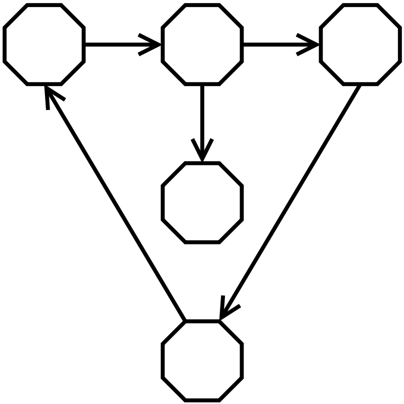
Optimal score

16

Algorithm score

9

Sample game



Spoiler

Ha! Looser!

Optimal score

16

Algorithm score

9

On-line results

Questions:

- What is the optimal strategy for both players?

On-line results

Questions:

- What is the optimal strategy for both players?
- In a graph of n vertices, what will be the outcome of such game, assuming both players play optimally?

On-line results

Questions:

- What is the optimal strategy for both players?
- In a graph of n vertices, what will be the outcome of such game, assuming both players play optimally?

Answers:

- A certain Algorithm player can guarantee himself at least $\Omega\left(n\frac{\log n}{\log\log n}\right)$ reachable pairs.

On-line results

Questions:

- What is the optimal strategy for both players?
- In a graph of n vertices, what will be the outcome of such game, assuming both players play optimally?

Answers:

- A certain Algorithm player can guarantee himself at least $\Omega\left(n\frac{\log n}{\log\log n}\right)$ reachable pairs.
- Spoiler has a strategy of giving vertices and edges such that this number will always be bounded by $O\left(n\frac{\log n}{\log\log n}\right)$.

Greedy Algorithm

Suppose that the graph is a tree. In each round we are given vertex s with one edge (s, t).

Greedy Algorithm

Suppose that the graph is a tree. In each round we are given vertex s with one edge (s, t).

- t_{out} $:=$ the number of vertices reachable from t
- t_{in} $:=$ the number of vertices from which t is reachable
- Choose direction $s \rightarrow t$ if t_{out} is larger, $t \rightarrow s$ otherwise

Sketch of proof

- Let $order(s) = max(t_{out}, t_{in}) + 1$
- The smaller of t_{out}, t_{in} goes up every time a new vertex is connected to t
- A vertex can have at most k children of order k.
- There are at most $(k + 2)!$ vertices of order k.

Sketch of proof

- Let $order(s) = max(t_{out}, t_{in}) + 1$
- The smaller of t_{out}, t_{in} goes up every time a new vertex is connected to t
- A vertex can have at most k children of order k.
- There are at most $(k+2)!$ vertices of order k.

Corollary: The total number of connected pairs is $\Omega\left(n \frac{\log n}{\log \log n}\right)$.

Sketch of proof

- Let $order(s) = max(t_{out}, t_{in}) + 1$
- The smaller of t_{out}, t_{in} goes up every time a new vertex is connected to t
- A vertex can have at most k children of order k.
- There are at most $(k+2)!$ vertices of order k.

Corollary: The total number of connected pairs is $\Omega\left(n\frac{\log n}{\log\log n}\right)$.

The same proof works for general graphs.

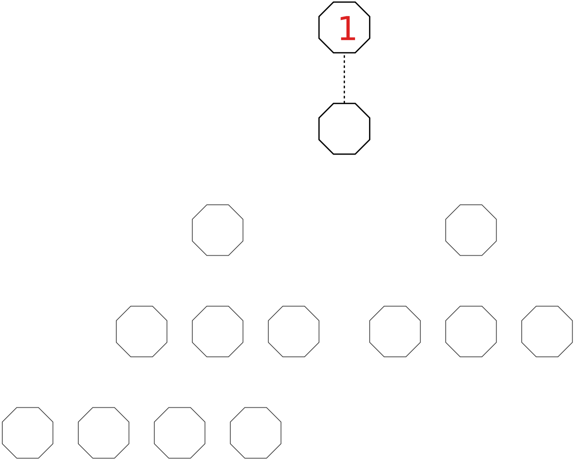
Factorial tree Strategy

- start with single node of rank 1

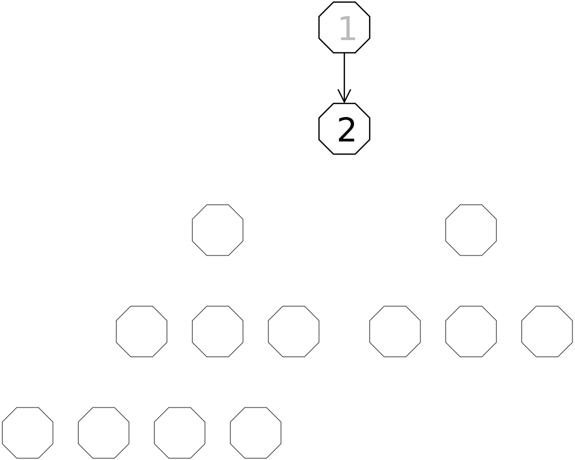
Factorial tree Strategy

- start with single node of rank 1
- choose a leaf of lowest rank r
- attach r children of rank $r + 1$

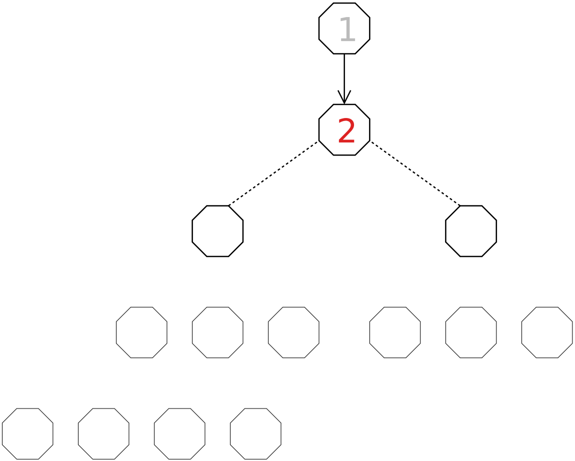
Factorial tree Strategy vs. Greedy Algorithm



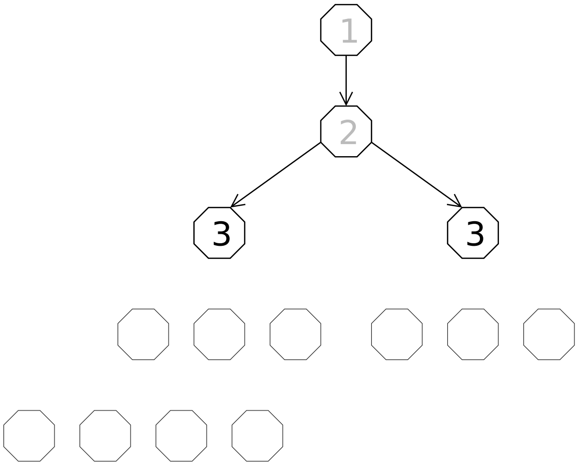
Factorial tree Strategy vs. Greedy Algorithm



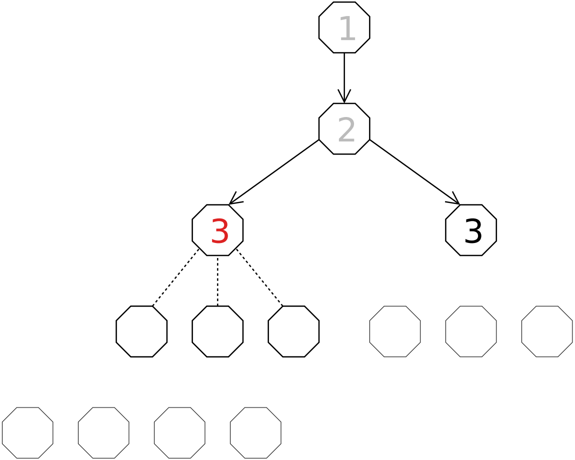
Factorial tree Strategy vs. Greedy Algorithm



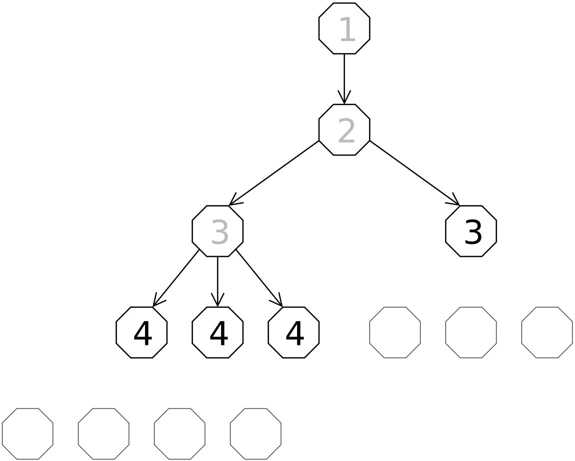
Factorial tree Strategy vs. Greedy Algorithm



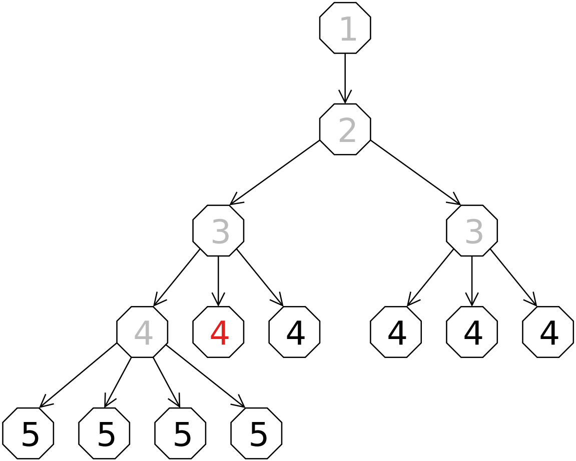
Factorial tree Strategy vs. Greedy Algorithm



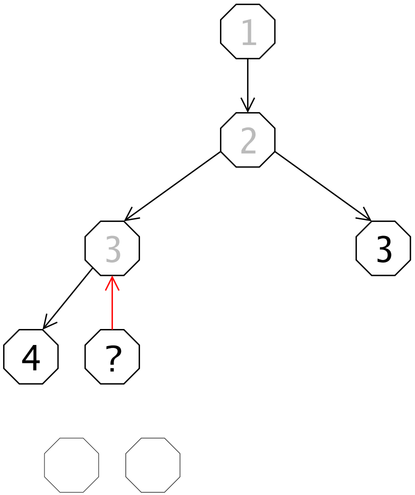
Factorial tree Strategy vs. Greedy Algorithm



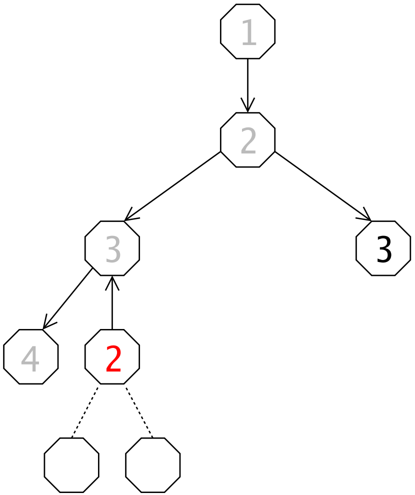
Factorial tree Strategy vs. Greedy Algorithm



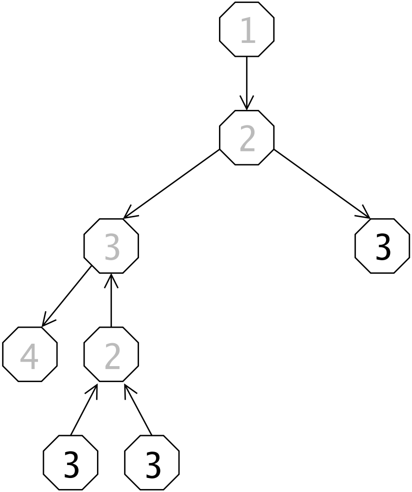
Non-greedy opponent



Non-greedy opponent



Non-greedy opponent



Sketch of proof

- The number of connected pairs is bounded by the sum of ranks
- If there is a vertex of rank r there are at least $(r-2)!$ vertices

Sketch of proof

- The number of connected pairs is bounded by the sum of ranks
- If there is a vertex of rank r there are at least $(r-2)!$ vertices

Corollary: The total number of connected pairs is $O\left(n\frac{\log n}{\log\log n}\right)$.

Summary

- Optimal players achieve $\Theta\left(n\frac{\log n}{\log\log n}\right)$ connected pairs
- This is poor, compared to $\Omega\left(n^2\right)$ in the off-line case
- In the game defined, Spoiler always should construct a tree