On-line reachability maximization problem

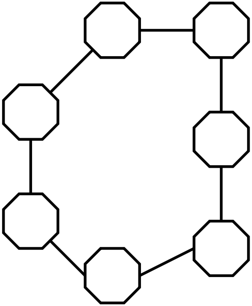
Lech Duraj

Theoretical Computer Science Department
Jagiellonian University

CLA 2009

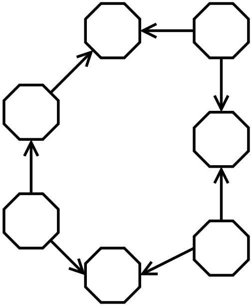
Building a one-way network

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



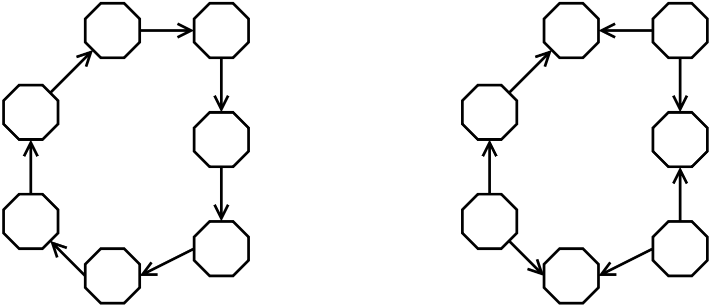
Building a one-way network

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, solvable in polynomial time.

Off-line results

- For trees, reachable pairs and average connectivity are the same problem, solvable in polynomial time.
- The algorithm can be adapted for weighted trees, and thus for general graphs (Hakimi, Schmeichel, Young, 1997).

Off-line results

- For trees, reachable pairs and average connectivity are the same problem, solvable in polynomial time.
- The algorithm can be adapted for weighted trees, and thus for general graphs (Hakimi, Schmeichel, Young, 1997).
- The optimal solution always gives $\Theta(n^2)$ connected pairs.

Off-line results

- For trees, reachable pairs and average connectivity are the same problem, solvable in polynomial time.
- The algorithm can be adapted for weighted trees, and thus for general graphs (Hakimi, Schmeichel, Young, 1997).
- The optimal solution always gives $\Theta(n^2)$ connected pairs.
- It is unknown if average connectivity problem is NP-hard.

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

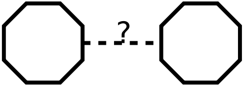
Constraint: graph is connected
Goal: minimize the number of connected pairs

Algorithm

- directs new edges
- decisions are permanent

Goal: maximize the number of connected pairs

Sample game



Spoiler

starts with a single edge

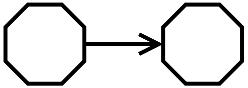
Optimal score

1

Algorithm score

$0+?$

Sample game



Algorithm

directs the edge

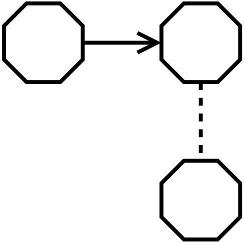
Optimal score

1

Algorithm score

1

Sample game



Spoiler

adds another edge

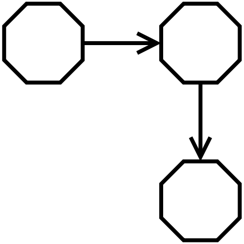
Optimal score

3

Algorithm score

$1+?$

Sample game



Algorithm

directs the edge

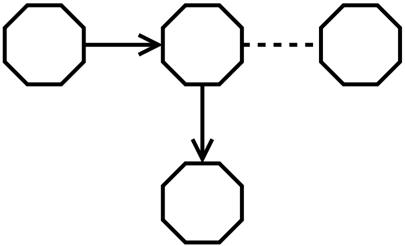
Optimal score

3

Algorithm score

3

Sample game



Spoiler

adds another edge

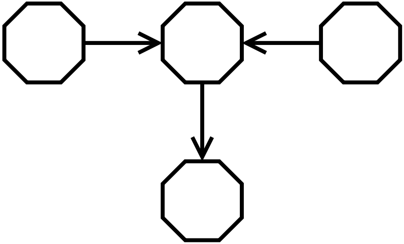
Optimal score

5

Algorithm score

3+?

Sample game



Algorithm

directs the edge

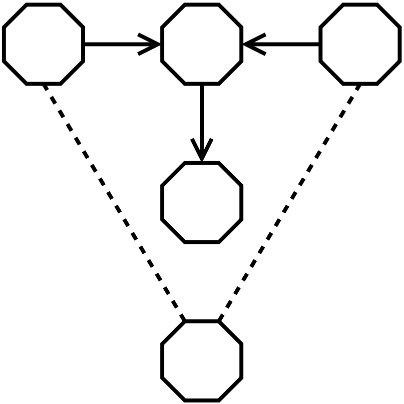
Optimal score

5

Algorithm score

5

Sample game



Spoiler

adds two edges

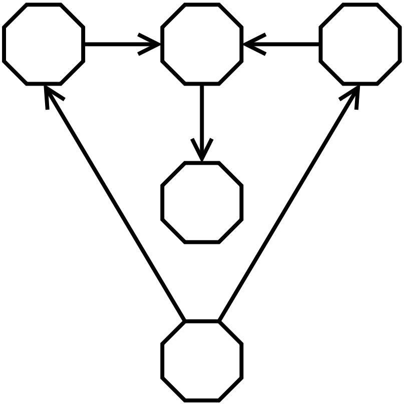
Optimal score

16

Algorithm score

$5+?$

Sample game



Algorithm

can't achieve optimum

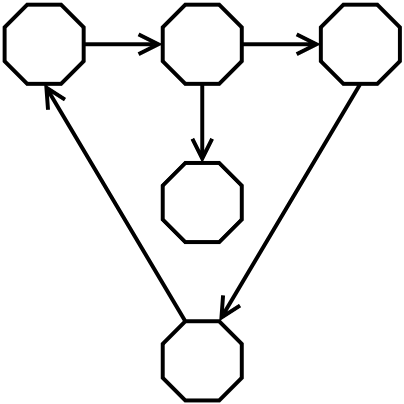
Optimal score

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

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 analysis

- Let $order(s) = max(t_{out}, t_{in})$
- 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 $\Omega(n\frac{\log n}{\log \log n})$.

Sketch of analysis

- Let $order(s) = max(t_{out}, t_{in})$
- 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 $\Omega(n\frac{\log n}{\log\log n})$.
The same proof works for general graphs.

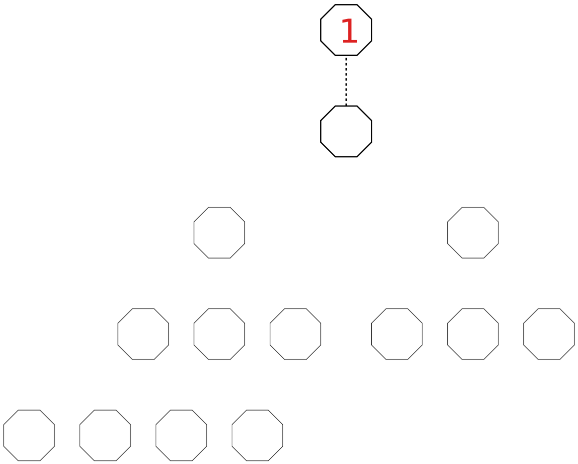
Factorial tree Strategy

- start with single node of rank 1

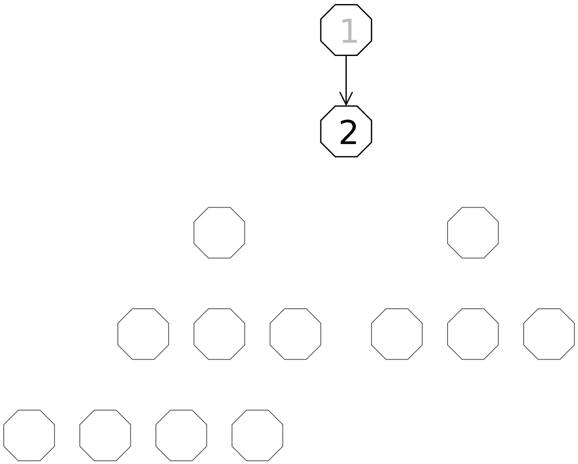
Factorial tree Strategy

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

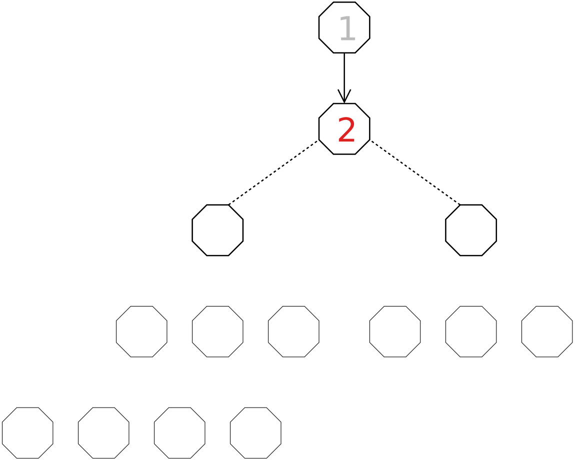
Factorial tree Strategy vs. Greedy Algorithm



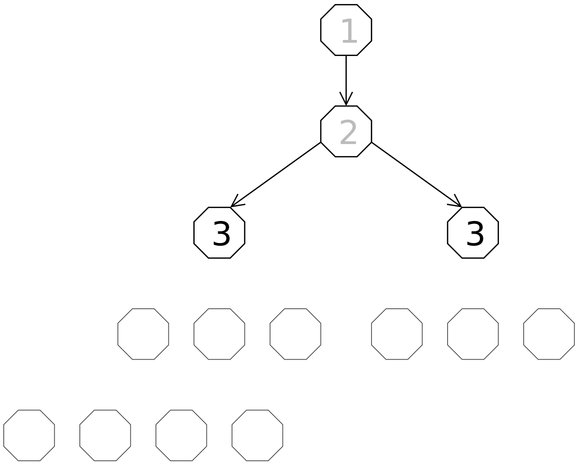
Factorial tree Strategy vs. Greedy Algorithm



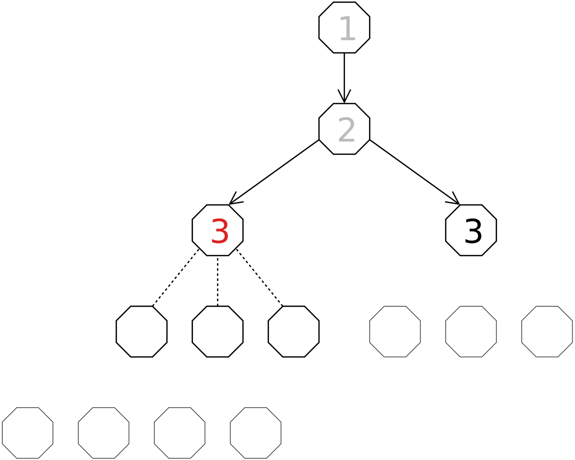
Factorial tree Strategy vs. Greedy Algorithm



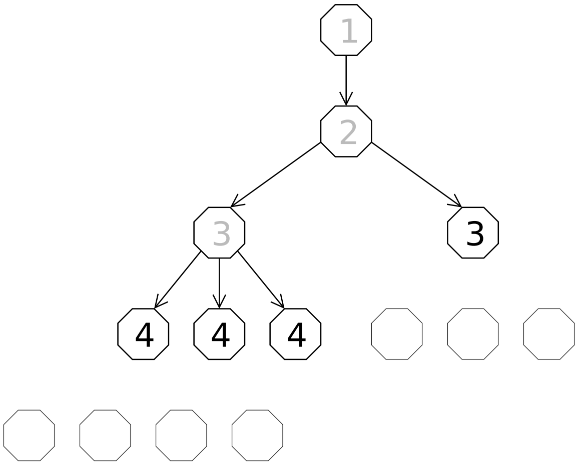
Factorial tree Strategy vs. Greedy Algorithm



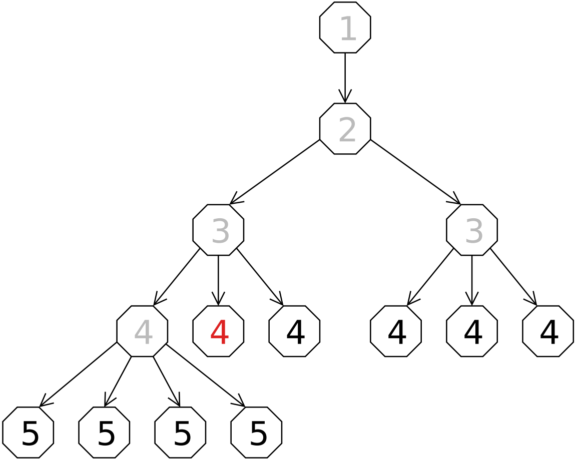
Factorial tree Strategy vs. Greedy Algorithm



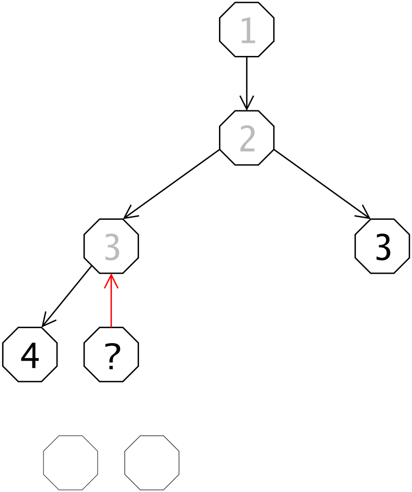
Factorial tree Strategy vs. Greedy Algorithm



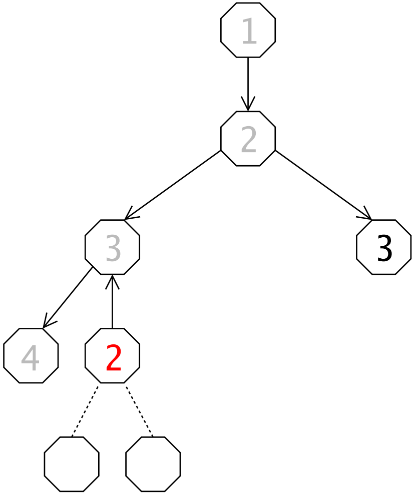
Factorial tree Strategy vs. Greedy Algorithm



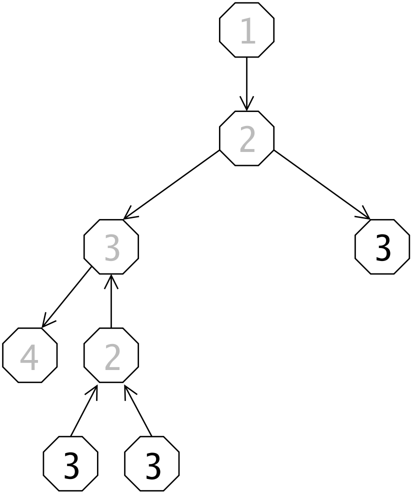
Non-greedy opponent



Non-greedy opponent



Non-greedy opponent



Sketch of analysis

- 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 $O(n\frac{\log n}{\log\log n})$.

Summary

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

Slightly different game

- **Bounded degree**: Suppose that no vertex can have degree higher than d.

Slightly different game

- **Bounded degree**: Suppose that no vertex can have degree higher than d.
- The analysis is very similar: a vertex of order k can have no more than d children of order $k + 1$

Slightly different game

- **Bounded degree**: Suppose that no vertex can have degree higher than d.
- The analysis is very similar: a vertex of order k can have no more than d children of order $k+1$
- At least half of the vertices must have order at least $\Omega(\log n)$, so the total number of connected pairs is at least $\Omega(n \log n)$

Slightly different game

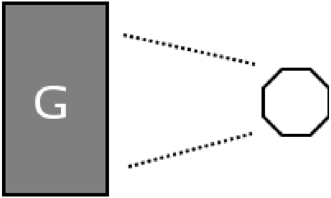
- **Bounded degree**: Suppose that no vertex can have degree higher than d.
- The analysis is very similar: a vertex of order k can have no more than d children of order $k+1$
- At least half of the vertices must have order at least $\Omega(\log n)$, so the total number of connected pairs is at least $\Omega(n \log n)$
- Instead of a factorial tree, Spoiler gives a binary tree, limiting Algorithm to $O(n \log n)$ score.

Easy game

- **Biconnected graph**: Starting from the 3rd turn, the graph should remain biconnected.

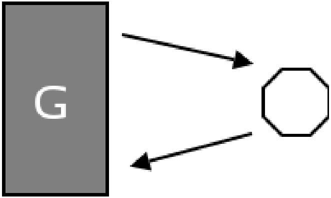
Easy game

- **Biconnected graph**: Starting from the 3rd turn, the graph should remain biconnected.
- However, every incoming vertex has degree at least 2...



Easy game

- **Biconnected graph**: Starting from the 3rd turn, the graph should remain biconnected.
- However, every incoming vertex has degree at least 2...
- ...so the algorithm is obvious.



More games

- P_k**-free graph**: The graph must not contain a k-vertex path without shortcuts.

More games

- P_k**-free graph**: The graph must not contain a k-vertex path without shortcuts.
- This means the graph has a spanning tree of low diameter and thus containing high degree vertices.

More games

- P_k**-free graph**: The graph must not contain a k-vertex path without shortcuts.
- This means the graph has a spanning tree of low diameter and thus containing high degree vertices.
- The score of this game is $\Omega(n^\alpha)$ for some $\alpha > 1$.

More games

- P_k**-free graph**: The graph must not contain a k-vertex path without shortcuts.
- This means the graph has a spanning tree of low diameter and thus containing high degree vertices.
- The score of this game is $\Omega(n^\alpha)$ for some $\alpha > 1$.
- A very similar **bounded diameter game** becomes very complicated – the Spoiler can add non-tree edges to decrease diameter during the game.