

# Crossover Can Provably Speed Up Evolutionary Computation

Benjamin Doerr, Edda Happ, Christian Klein

July 14, 2007

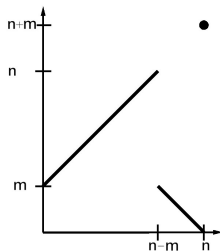
## Problem

- EAs typically use both mutation and crossover.
- Practise shows that crossover improves runtime.
- But: Little proof that crossover is useful.

## Pseudo-boolean *jump* Function

jump function  $j_m : \{0, 1\}^n \rightarrow \mathbb{R}$

$$j_m(x_1, \dots, x_n) = \begin{cases} m + \sum x_i & \text{if } \sum x_i \leq n - m \\ m + n & \text{if } \sum x_i = n \\ n - \sum x_i & \text{else} \end{cases}$$

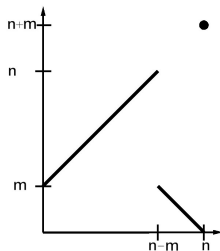


T. Jansen and I. Wegener. On the analysis of evolutionary algorithms - a proof that crossover really can help. *ESA 1999, Algorithmica 2002*

## Pseudo-boolean *jump* Function

jump function  $j_m : \{0, 1\}^n \rightarrow \mathbb{R}$

$$j_m(x_1, \dots, x_n) = \begin{cases} m + \sum x_i & \text{if } \sum x_i \leq n - m \\ m + n & \text{if } \sum x_i = n \\ n - \sum x_i & \text{else} \end{cases}$$



## Result

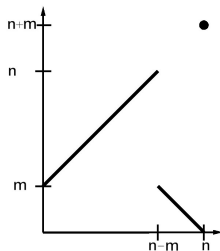
- Mutation only: Expected optimization time  $\Theta(n^m)$ .

T. Jansen and I. Wegener. On the analysis of evolutionary algorithms - a proof that crossover really can help. *ESA 1999, Algorithmica 2002*

## Pseudo-boolean *jump* Function

jump function  $j_m : \{0, 1\}^n \rightarrow \mathbb{R}$

$$j_m(x_1, \dots, x_n) = \begin{cases} m + \sum x_i & \text{if } \sum x_i \leq n - m \\ m + n & \text{if } \sum x_i = n \\ n - \sum x_i & \text{else} \end{cases}$$



## Result

- Mutation only: Expected optimization time  $\Theta(n^m)$ .
- Mutation and crossover: Expected optimization time  $O(n^2 \log n)$ .

T. Jansen and I. Wegener. On the analysis of evolutionary algorithms - a proof that crossover really can help. *ESA 1999, Algorithmica 2002*

## Simplified Ising Problem

Given a tree  $T = (V, E)$ . Assign  $x_i \in \{-1, +1\}$  to the vertices maximizing

$$\sum_{(v_i, v_j) \in E} x_i x_j.$$

## Simplified Ising Problem

Given a tree  $T = (V, E)$ . Assign  $x_i \in \{-1, +1\}$  to the vertices maximizing

$$\sum_{(v_i, v_j) \in E} x_i x_j.$$

► Find a monochromatic coloring!

## Simplified Ising Problem

Given a tree  $T = (V, E)$ . Assign  $x_i \in \{-1, +1\}$  to the vertices maximizing

$$\sum_{(v_i, v_j) \in E} x_i x_j.$$

► Find a monochromatic coloring!

## Result

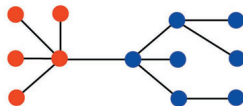
- Only mutation: Optimization time exponential.

## Simplified Ising Problem

Given a tree  $T = (V, E)$ . Assign  $x_i \in \{-1, +1\}$  to the vertices maximizing

$$\sum_{(v_i, v_j) \in E} x_i x_j.$$

► Find a monochromatic coloring!



## Result

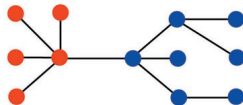
- Only mutation: Optimization time exponential.

## Simplified Ising Problem

Given a tree  $T = (V, E)$ . Assign  $x_i \in \{-1, +1\}$  to the vertices maximizing

$$\sum_{(v_i, v_j) \in E} x_i x_j.$$

► Find a monochromatic coloring!



## Result

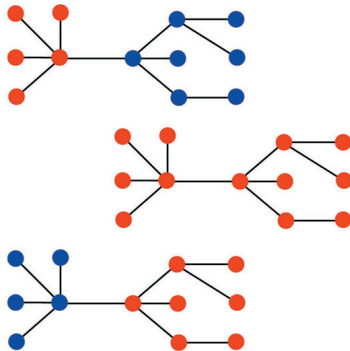
- Only mutation: Optimization time exponential.
- Mutation, fitness sharing and 1-point crossover:  $O(n^3)$  optimization time.

## Simplified Ising Problem

Given a tree  $T = (V, E)$ . Assign  $x_i \in \{-1, +1\}$  to the vertices maximizing

$$\sum_{(v_i, v_j) \in E} x_i x_j.$$

► Find a monochromatic coloring!



## Result

- Only mutation: Optimization time exponential.
- Mutation, fitness sharing and 1-point crossover:  $O(n^3)$  optimization time.

## Summary Existing Results

- Some proofs that crossover helps

## Summary Existing Results

- Some proofs that crossover helps
- Artificial problems only

## Summary Existing Results

- Some proofs that crossover helps
- Artificial problems only
- Some additional critique

## Summary Existing Results

- Some proofs that crossover helps
- Artificial problems only
- Some additional critique
  - $1 - o(1)$  fraction mutation

## Summary Existing Results

- Some proofs that crossover helps
- Artificial problems only
- Some additional critique
  - $1 - o(1)$  fraction mutation
  - need shared fitness

## Summary Existing Results

- Some proofs that crossover helps
- Artificial problems only
- Some additional critique
  - $1 - o(1)$  fraction mutation
  - need shared fitness

## Summary Existing Results

- Some proofs that crossover helps
- Artificial problems only
- Some additional critique
  - $1 - o(1)$  fraction mutation
  - need shared fitness

## Our Result

- Rigorous analysis of the All Pairs Shortest Path problem.

## Summary Existing Results

- Some proofs that crossover helps
- Artificial problems only
- Some additional critique
  - $1 - o(1)$  fraction mutation
  - need shared fitness

## Our Result

- Rigorous analysis of the All Pairs Shortest Path problem.
- Mutation only: Expected optimization time  $\Theta(n^4)$

# State-of-the-Art, our Results

## Summary Existing Results

- Some proofs that crossover helps
- Artificial problems only
- Some additional critique
  - $1 - o(1)$  fraction mutation
  - need shared fitness

## Our Result

- Rigorous analysis of the All Pairs Shortest Path problem.
- Mutation only: Expected optimization time  $\Theta(n^4)$
- Mutation and crossover: Exp. opt. time  $O(n^{3.5}\sqrt{\log n})$

# State-of-the-Art, our Results

## Summary Existing Results

- Some proofs that crossover helps
- Artificial problems only
- Some additional critique
  - $1 - o(1)$  fraction mutation
  - need shared fitness

## Our Result

- Rigorous analysis of the All Pairs Shortest Path problem.
- Mutation only: Expected optimization time  $\Theta(n^4)$
- Mutation and crossover: Exp. opt. time  $O(n^{3.5}\sqrt{\log n})$

## Remainder of the talk

# State-of-the-Art, our Results

## Summary Existing Results

- Some proofs that crossover helps
- Artificial problems only
- Some additional critique
  - $1 - o(1)$  fraction mutation
  - need shared fitness

## Our Result

- Rigorous analysis of the All Pairs Shortest Path problem.
- Mutation only: Expected optimization time  $\Theta(n^4)$
- Mutation and crossover: Exp. opt. time  $O(n^{3.5}\sqrt{\log n})$

## Remainder of the talk

- All Pairs Shortest Path problem

# State-of-the-Art, our Results

## Summary Existing Results

- Some proofs that crossover helps
- Artificial problems only
- Some additional critique
  - $1 - o(1)$  fraction mutation
  - need shared fitness

## Our Result

- Rigorous analysis of the All Pairs Shortest Path problem.
- Mutation only: Expected optimization time  $\Theta(n^4)$
- Mutation and crossover: Exp. opt. time  $O(n^{3.5}\sqrt{\log n})$

## Remainder of the talk

- All Pairs Shortest Path problem
- A  $(\mu + 1)$  evolutionary algorithm for the APSP

# State-of-the-Art, our Results

## Summary Existing Results

- Some proofs that crossover helps
- Artificial problems only
- Some additional critique
  - $1 - o(1)$  fraction mutation
  - need shared fitness

## Our Result

- Rigorous analysis of the All Pairs Shortest Path problem.
- Mutation only: Expected optimization time  $\Theta(n^4)$
- Mutation and crossover: Exp. opt. time  $O(n^{3.5}\sqrt{\log n})$

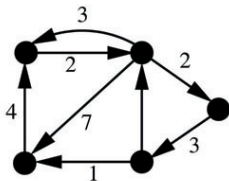
## Remainder of the talk

- All Pairs Shortest Path problem
- A  $(\mu + 1)$  evolutionary algorithm for the APSP
- Some proof ideas.

# The All Pairs Shortest Path (APSP) Problem

## Given

- Directed graph  $G = (V, E)$ .
- edge weights  $w : E \rightarrow \mathbb{Z} \cup \{\infty\}$
- No negative cycles!



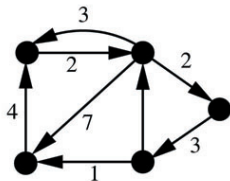
# The All Pairs Shortest Path (APSP) Problem

## Given

- Directed graph  $G = (V, E)$ .
- edge weights  $w : E \rightarrow \mathbb{Z} \cup \{\infty\}$
- No negative cycles!

## Problem

- For all pairs  $(u, v) \in V \times V$ , find a shortest path (SP) from  $u$  to  $v$ .



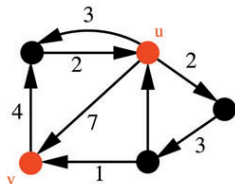
# The All Pairs Shortest Path (APSP) Problem

## Given

- Directed graph  $G = (V, E)$ .
- edge weights  $w : E \rightarrow \mathbb{Z} \cup \{\infty\}$
- No negative cycles!

## Problem

- For all pairs  $(u, v) \in V \times V$ , find a shortest path (SP) from  $u$  to  $v$ .



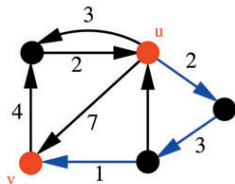
# The All Pairs Shortest Path (APSP) Problem

## Given

- Directed graph  $G = (V, E)$ .
- edge weights  $w : E \rightarrow \mathbb{Z} \cup \{\infty\}$
- No negative cycles!

## Problem

- For all pairs  $(u, v) \in V \times V$ , find a shortest path (SP) from  $u$  to  $v$ .



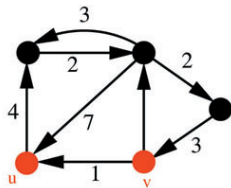
# The All Pairs Shortest Path (APSP) Problem

## Given

- Directed graph  $G = (V, E)$ .
- edge weights  $w : E \rightarrow \mathbb{Z} \cup \{\infty\}$
- No negative cycles!

## Problem

- For all pairs  $(u, v) \in V \times V$ , find a shortest path (SP) from  $u$  to  $v$ .



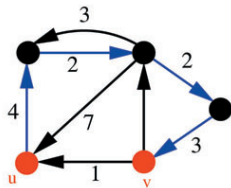
# The All Pairs Shortest Path (APSP) Problem

## Given

- Directed graph  $G = (V, E)$ .
- edge weights  $w : E \rightarrow \mathbb{Z} \cup \{\infty\}$
- No negative cycles!

## Problem

- For all pairs  $(u, v) \in V \times V$ , find a shortest path (SP) from  $u$  to  $v$ .



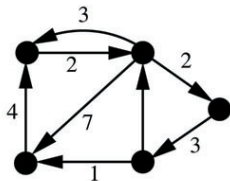
# The All Pairs Shortest Path (APSP) Problem

## Given

- Directed graph  $G = (V, E)$ .
- edge weights  $w : E \rightarrow \mathbb{Z} \cup \{\infty\}$
- No negative cycles!

## Problem

- For all pairs  $(u, v) \in V \times V$ , find a shortest path (SP) from  $u$  to  $v$ .
- In total:  $\mu = n(n - 1)$  non-trivial SPs.



# The All Pairs Shortest Path (APSP) Problem

## Given

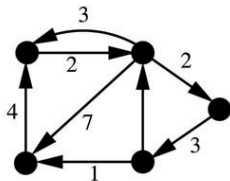
- Directed graph  $G = (V, E)$ .
- edge weights  $w : E \rightarrow \mathbb{Z} \cup \{\infty\}$
- No negative cycles!

## Problem

- For all pairs  $(u, v) \in V \times V$ , find a shortest path (SP) from  $u$  to  $v$ .
- In total:  $\mu = n(n - 1)$  non-trivial SPs.

## Classical Results ( $|V| = n, |E| = m$ )

- Floyd-Warshall algorithm:  $O(n^3)$ .
- Johnson's algorithm:  $O(n^2 \log n + nm)$ .



# A $(\mu + 1)$ Evolutionary Algorithm for the APSP

## Individuals and Fitness Function

- Individuals are simple paths:  $I = (e_1, \dots, e_k)$  with  $e_i \in E$ .
- Fitness of individual  $I$ : Length of path

$$f(I) = \sum_{i=1}^k w(e_i)$$

# A $(\mu + 1)$ Evolutionary Algorithm for the APSP

## Individuals and Fitness Function

- Individuals are simple paths:  $I = (e_1, \dots, e_k)$  with  $e_i \in E$ .
- Fitness of individual  $I$ : Length of path

$$f(I) = \sum_{i=1}^k w(e_i)$$

## Population and Selection

- Initial population:  $\mathcal{I} := \{(e) \mid e \in E\}$ .
- At most one SP for each vertex pair (diversity mechanism).

# A $(\mu + 1)$ Evolutionary Algorithm for the APSP

## Individuals and Fitness Function

- Individuals are simple paths:  $I = (e_1, \dots, e_k)$  with  $e_i \in E$ .
- Fitness of individual  $I$ : Length of path

$$f(I) = \sum_{i=1}^k w(e_i)$$

## Population and Selection

- Initial population:  $\mathcal{I} := \{(e) \mid e \in E\}$ .
- At most one SP for each vertex pair (diversity mechanism).
  - ▶ At most  $\mu = n(n - 1)$  individuals.

# A $(\mu + 1)$ Evolutionary Algorithm for the APSP

## Individuals and Fitness Function

- Individuals are simple paths:  $I = (e_1, \dots, e_k)$  with  $e_i \in E$ .
- Fitness of individual  $I$ : Length of path

$$f(I) = \sum_{i=1}^k w(e_i)$$

## Population and Selection

- Initial population:  $\mathcal{I} := \{(e) \mid e \in E\}$ .
- At most one SP for each vertex pair (diversity mechanism).
  - ▶ At most  $\mu = n(n-1)$  individuals.
- Selection: Replace an existing path  $I$  from  $u$  to  $v$  by a newly generated one  $I'$  if  $f(I') \leq f(I)$ .

# A $(\mu + 1)$ Evolutionary Algorithm for the APSP

## Algorithm

$\mathcal{I} := \{(e) \mid e \in E\}$

# A $(\mu + 1)$ Evolutionary Algorithm for the APSP

## Algorithm

$\mathcal{I} := \{(e) \mid e \in E\}$

repeat

forever

# A $(\mu + 1)$ Evolutionary Algorithm for the APSP

## Algorithm

$\mathcal{I} := \{(e) \mid e \in E\}$

repeat

    either

        do crossover

forever

# A $(\mu + 1)$ Evolutionary Algorithm for the APSP

## Algorithm

$\mathcal{I} := \{(e) \mid e \in E\}$

repeat

    either

        do crossover

        or do mutation

forever

# A $(\mu + 1)$ Evolutionary Algorithm for the APSP

## Algorithm

```
 $\mathcal{I} := \{(e) \mid e \in E\}$   
repeat  
  either  
    do crossover  
    or do mutation  
  if  $f(I') \leq f(I)$   
    then replace  $I$  by  $I'$  in  $\mathcal{I}$   
forever
```

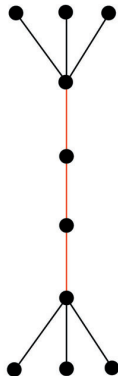
## Single Mutation of an Individual $I$



# Mutation Operator

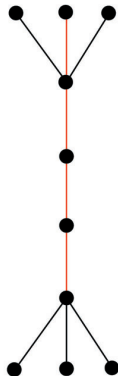
## Single Mutation of an Individual $I$

- Pick an edge  $e$  u.a.r. from all edges incident with an endvertex of  $I$



## Single Mutation of an Individual $I$

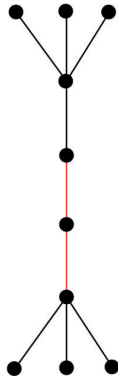
- Pick an edge  $e$  u.a.r. from all edges incident with an endvertex of  $I$ 
  - Add  $e$  to  $I$ , or



# Mutation Operator

## Single Mutation of an Individual $I$

- Pick an edge  $e$  u.a.r. from all edges incident with an endvertex of  $I$ 
  - Add  $e$  to  $I$ , or
  - delete  $e$  from  $I$ , if  $e$  is already on the path  $I$ .



# Mutation Operator

## Single Mutation of an Individual $I$

- Pick an edge  $e$  u.a.r. from all edges incident with an endvertex of  $I$ 
  - Add  $e$  to  $I$ , or
  - delete  $e$  from  $I$ , if  $e$  is already on the path  $I$ .

## Mutation Operator

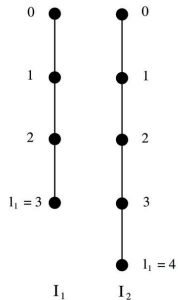
- Pick  $I$  u.a.r. from  $\mathcal{I}$ .
- Pick  $s$  at random according to  $\text{Pois}(\lambda = 1)$ .
- Sequentially perform  $s + 1$  single mutations.



# Crossover Operator

## Crossover Operator

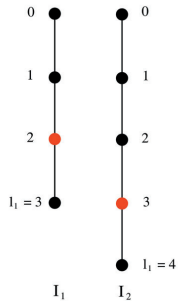
- Pick  $I_1, I_2$  u.a.r. from  $\mathcal{I}$ . Let  $\ell_1 = |I_1|, \ell_2 = |I_2|$ .



# Crossover Operator

## Crossover Operator

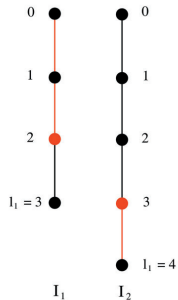
- Pick  $l_1, l_2$  u.a.r. from  $\mathcal{I}$ . Let  $\ell_1 = |l_1|, \ell_2 = |l_2|$ .
- Pick  $i \in [0..\ell_1], j \in [0..\ell_2]$  u.a.r.



# Crossover Operator

## Crossover Operator

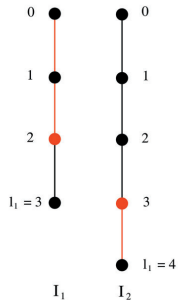
- Pick  $I_1, I_2$  u.a.r. from  $\mathcal{I}$ . Let  $\ell_1 = |I_1|, \ell_2 = |I_2|$ .
- Pick  $i \in [0..\ell_1], j \in [0..\ell_2]$  u.a.r.
- New individual  $I'$  contains first  $i$  edges from  $I_1$  and last  $\ell_2 - j$  edges from  $I_2$ .



# Crossover Operator

## Crossover Operator

- Pick  $I_1, I_2$  u.a.r. from  $\mathcal{I}$ . Let  $\ell_1 = |I_1|, \ell_2 = |I_2|$ .
- Pick  $i \in [0.. \ell_1], j \in [0.. \ell_2]$  u.a.r.
- New individual  $I'$  contains first  $i$  edges from  $I_1$  and last  $\ell_2 - j$  edges from  $I_2$ .
- Note: Often,  $I'$  is not a path. Then it never enters the population.



# Summary: The $(\mu + 1)$ Evolutionary Algorithm for the APSP

## Algorithm

$\mathcal{I} := \{(e) \mid e \in E\}$

# Summary: The $(\mu + 1)$ Evolutionary Algorithm for the APSP

## Algorithm

$\mathcal{I} := \{(e) \mid e \in E\}$

repeat

    either do crossover or do mutation

    if  $f(I') \leq f(I)$

        then replace  $I$  by  $I'$  in  $\mathcal{I}$

forever

# Summary: The $(\mu + 1)$ Evolutionary Algorithm for the APSP

## Algorithm

$\mathcal{I} := \{(e) \mid e \in E\}$

repeat

    either do crossover or do mutation

        Mutation: Add/delete  $\text{Pois}(\lambda = 1)$  random edges to random individual.

        Crossover: Concatenate a random initial and terminal segment.

    if  $f(I') \leq f(I)$

        then replace  $I$  by  $I'$  in  $\mathcal{I}$

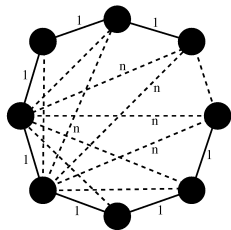
forever

# Only Mutation: Expected Optimization Time $\Omega(n^4)$

## Idea

- Consider  $K_n$  with

$$w(v_i, v_j) = \begin{cases} 1 & \text{if } i + 1 = j, \\ n & \text{else.} \end{cases}$$



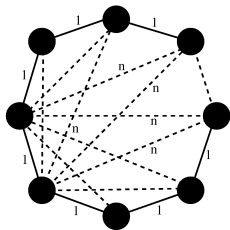
# Only Mutation: Expected Optimization Time $\Omega(n^4)$

## Idea

- Consider  $K_n$  with

$$w(v_i, v_j) = \begin{cases} 1 & \text{if } i + 1 = j, \\ n & \text{else.} \end{cases}$$

- Claim: One needs  $\Omega(n^4)$  time to find  $I = ((v_1, v_2), \dots, (v_{n-1}, v_n))$ .



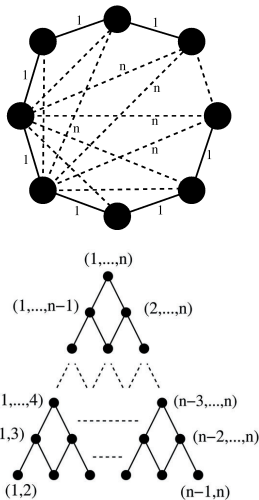
# Only Mutation: Expected Optimization Time $\Omega(n^4)$

## Idea

- Consider  $K_n$  with

$$w(v_i, v_j) = \begin{cases} 1 & \text{if } i + 1 = j, \\ n & \text{else.} \end{cases}$$

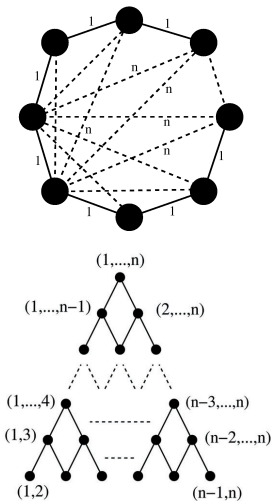
- Claim: One needs  $\Omega(n^4)$  time to find  $I = ((v_1, v_2), \dots, (v_{n-1}, v_n))$ .
- Finding  $I$  means going from a leaf to  $I$  ("trail in auxiliary graph").



# Only Mutation: Expected Optimization Time $\Omega(n^4)$

## Idea

- Consider  $K_n$  with
$$w(v_i, v_j) = \begin{cases} 1 & \text{if } i + 1 = j, \\ n & \text{else.} \end{cases}$$
- Claim: One needs  $\Omega(n^4)$  time to find  $I = ((v_1, v_2), \dots, (v_{n-1}, v_n))$ .
- Finding  $I$  means going from a leaf to  $I$  ("trail in auxiliary graph").
  - ▶ Fixed trail of length  $\ell$ :
$$\Pr[\text{go trail in } \leq 10n^4 \text{ steps}] \leq \frac{1}{5^\ell}.$$



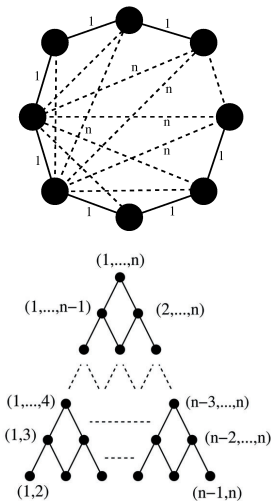
# Only Mutation: Expected Optimization Time $\Omega(n^4)$

## Idea

- Consider  $K_n$  with

$$w(v_i, v_j) = \begin{cases} 1 & \text{if } i + 1 = j, \\ n & \text{else.} \end{cases}$$

- Claim: One needs  $\Omega(n^4)$  time to find  $I = ((v_1, v_2), \dots, (v_{n-1}, v_n))$ .
- Finding  $I$  means going from a leaf to  $I$  ("trail in auxiliary graph").
  - Fixed trail of length  $\ell$ :  
 $\Pr[\text{go trail in } \leq 10n^4 \text{ steps}] \leq \frac{1}{5^\ell}$ .
  - At most  $4^\ell$  trails of length  $\ell$ .



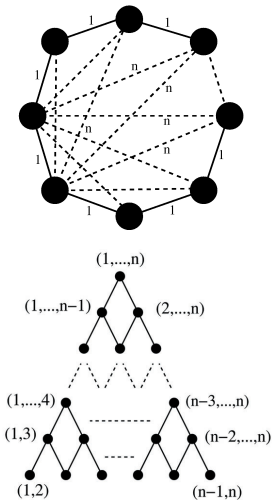
# Only Mutation: Expected Optimization Time $\Omega(n^4)$

## Idea

- Consider  $K_n$  with

$$w(v_i, v_j) = \begin{cases} 1 & \text{if } i + 1 = j, \\ n & \text{else.} \end{cases}$$

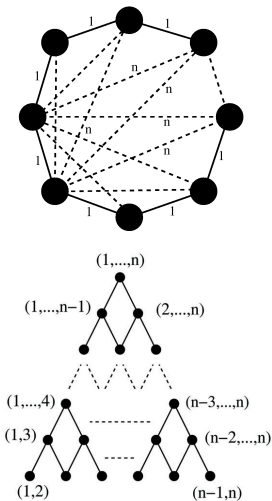
- Claim: One needs  $\Omega(n^4)$  time to find  $I = ((v_1, v_2), \dots, (v_{n-1}, v_n))$ .
- Finding  $I$  means going from a leaf to  $I$  ("trail in auxiliary graph").
  - Fixed trail of length  $\ell$ :  
 $\Pr[\text{go trail in } \leq 10n^4 \text{ steps}] \leq \frac{1}{5^\ell}$ .
  - At most  $4^\ell$  trails of length  $\ell$ .
- Summing over all  $\ell \geq n - 2$ :



# Only Mutation: Expected Optimization Time $\Omega(n^4)$

## Idea

- Consider  $K_n$  with
$$w(v_i, v_j) = \begin{cases} 1 & \text{if } i + 1 = j, \\ n & \text{else.} \end{cases}$$
- Claim: One needs  $\Omega(n^4)$  time to find  $I = ((v_1, v_2), \dots, (v_{n-1}, v_n))$ .
- Finding  $I$  means going from a leaf to  $I$  (“trail in auxiliary graph”).
  - ▶ Fixed trail of length  $\ell$ :
$$\Pr[\text{go trail in } \leq 10n^4 \text{ steps}] \leq \frac{1}{5^\ell}.$$
  - ▶ At most  $4^\ell$  trails of length  $\ell$ .
- Summing over all  $\ell \geq n - 2$ :
  - ▶  $\Pr[\text{Find } I \text{ in } \leq 10n^4 \text{ steps}] \leq \sum_{\ell=n-2}^{\infty} \frac{4^\ell}{5^\ell} = 5(\frac{4}{5})^{n-2} = \text{unlikely.}$

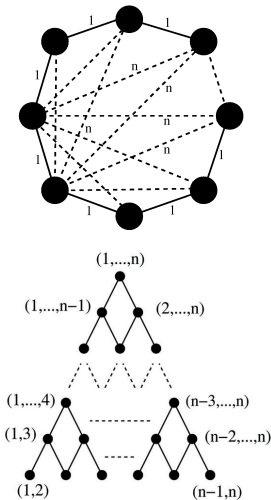


# Only Mutation: Expected Optimization Time $\Omega(n^4)$

## Idea

- Consider  $K_n$  with
$$w(v_i, v_j) = \begin{cases} 1 & \text{if } i + 1 = j, \\ n & \text{else.} \end{cases}$$
- Claim: One needs  $\Omega(n^4)$  time to find  $I = ((v_1, v_2), \dots, (v_{n-1}, v_n))$ .
- Finding  $I$  means going from a leaf to  $I$  (“trail in auxiliary graph”).
  - ▶ Fixed trail of length  $\ell$ :
$$\Pr[\text{go trail in } \leq 10n^4 \text{ steps}] \leq \frac{1}{5^\ell}.$$
  - ▶ At most  $4^\ell$  trails of length  $\ell$ .
- Summing over all  $\ell \geq n - 2$ :
  - ▶  $\Pr[\text{Find } I \text{ in } \leq 10n^4 \text{ steps}] \leq \sum_{\ell=n-2}^{\infty} \frac{4^\ell}{5^\ell} = 5(\frac{4}{5})^{n-2} = \text{unlikely.}$

$\Rightarrow \Omega(n^4)$  optimization time



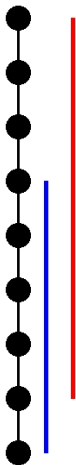
# Why are we Faster with Crossover?

Basics: Obtaining a SP  $P$  from two sub-paths



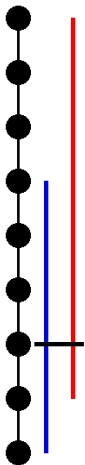
# Why are we Faster with Crossover?

Basics: Obtaining a SP  $P$  from two sub-paths



# Why are we Faster with Crossover?

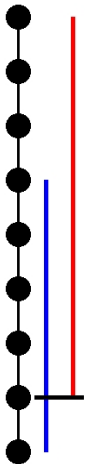
Basics: Obtaining a SP  $P$  from two sub-paths



# Why are we Faster with Crossover?

## Basics: Obtaining a SP $P$ from two sub-paths

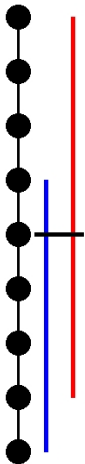
- There are different possible crossover positions.



# Why are we Faster with Crossover?

## Basics: Obtaining a SP $P$ from two sub-paths

- There are different possible crossover positions.



# Why are we Faster with Crossover?

## Basics: Obtaining a SP $P$ from two sub-paths

- There are different possible crossover positions.
- There are different possible sub-paths.



# Why are we Faster with Crossover?

## Basics: Obtaining a SP $P$ from two sub-paths

- There are different possible crossover positions.
- There are different possible sub-paths.



# Why are we Faster with Crossover?

## Basics: Obtaining a SP $P$ from two sub-paths

- There are different possible crossover positions.
- There are different possible sub-paths.

## Proof Outline

- Theorem: If all SPs having  $\leq k$  edges are in the population, then all SPs having  $\leq \frac{3}{2}k$  edges can be found in  $O\left(\frac{n^4 \log n}{k}\right)$  crossover steps.



# Why are we Faster with Crossover?

## Basics: Obtaining a SP $P$ from two sub-paths

- There are different possible crossover positions.
- There are different possible sub-paths.

## Proof Outline

- Theorem: If all SPs having  $\leq k$  edges are in the population, then all SPs having  $\leq \frac{3}{2}k$  edges can be found in  $O\left(\frac{n^4 \log n}{k}\right)$  crossover steps.
- First  $O(n^{3.5} \sqrt{\log n})$  steps:  
Mutation finds all SPs having  $< \sqrt{n \log(n)}$  edges.



# Why are we Faster with Crossover?

## Basics: Obtaining a SP $P$ from two sub-paths

- There are different possible crossover positions.
- There are different possible sub-paths.

## Proof Outline

- Theorem: If all SPs having  $\leq k$  edges are in the population, then all SPs having  $\leq \frac{3}{2}k$  edges can be found in  $O(\frac{n^4 \log n}{k})$  crossover steps.
- First  $O(n^{3.5} \sqrt{\log n})$  steps:  
Mutation finds all SPs having  $< \sqrt{n \log(n)}$  edges.
- Next  $O(n^{3.5} \sqrt{\log n})$  steps:  
Crossover finds all shortest paths.



# Why are we Faster with Crossover?

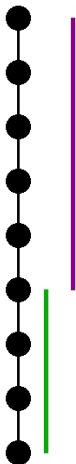
## Basics: Obtaining a SP $P$ from two sub-paths

- There are different possible crossover positions.
- There are different possible sub-paths.

## Proof Outline

- Theorem: If all SPs having  $\leq k$  edges are in the population, then all SPs having  $\leq \frac{3}{2}k$  edges can be found in  $O\left(\frac{n^4 \log n}{k}\right)$  crossover steps.
- First  $O(n^{3.5} \sqrt{\log n})$  steps:  
Mutation finds all SPs having  $< \sqrt{n \log(n)}$  edges.
- Next  $O(n^{3.5} \sqrt{\log n})$  steps:  
Crossover finds all shortest paths.

In total:  $O(n^{3.5} \sqrt{\log n})$  optimization time.



## Summary

- Previous work: Artificial examples show the use of crossover.

## Summary

- Previous work: Artificial examples show the use of crossover.
- Our result: A natural  $(\mu + 1)$ -EA for the APSP problem needs

## Summary

- Previous work: Artificial examples show the use of crossover.
- Our result: A natural  $(\mu + 1)$ -EA for the APSP problem needs
  - $\Omega(n^4)$  time using mutation only,

## Summary

- Previous work: Artificial examples show the use of crossover.
- Our result: A natural  $(\mu + 1)$ -EA for the APSP problem needs
  - $\Omega(n^4)$  time using mutation only,
  - $O(n^{3.5}\sqrt{\log n})$  time using mutation and crossover.

## Summary

- Previous work: Artificial examples show the use of crossover.
- Our result: A natural  $(\mu + 1)$ -EA for the APSP problem needs
  - $\Omega(n^4)$  time using mutation only,
  - $O(n^{3.5}\sqrt{\log n})$  time using mutation and crossover.

## Open Problem:

Find classical problems with larger gap!

## Summary

- Previous work: Artificial examples show the use of crossover.
- Our result: A natural  $(\mu + 1)$ -EA for the APSP problem needs
  - $\Omega(n^4)$  time using mutation only,
  - $O(n^{3.5}\sqrt{\log n})$  time using mutation and crossover.

## Open Problem:

Find classical problems with larger gap!

Thank You for Your Attention!