Computational Complexity of Evolutionary Computation in Combinatorial Optimisation

Frank Neumann[1] Carsten Witt[2]

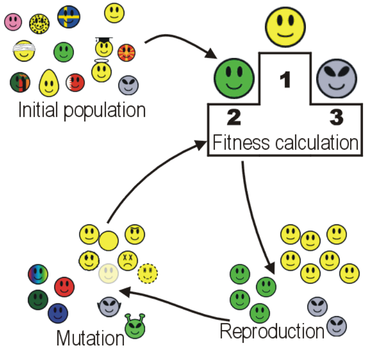
[1]Algorithms and Complexity Group
Max-Planck-Institute for Informatics
Saarbrücken, Germany

[2]Fakultät für Informatik, LS 2
Technische Universität Dortmund
Germany

Tutorial at PPSN 2008
14 September 2008

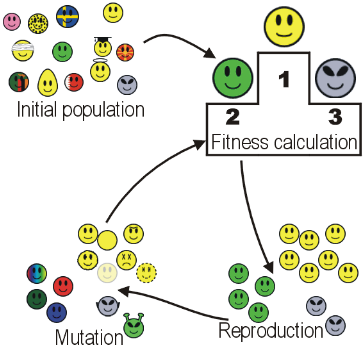
Most famous search heuristic: Evolutionary Algorithms (EAs)

- a bio-inspired heuristic
- paradigm: evolution in nature, "survival of the fittest"



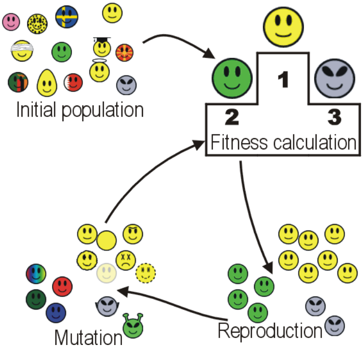
Most famous search heuristic: Evolutionary Algorithms (EAs)

- a bio-inspired heuristic
- paradigm: evolution in nature, "survival of the fittest"
- actually it's only an algorithm, a randomised search heuristic (RSH)



Most famous search heuristic: Evolutionary Algorithms (EAs)

- a bio-inspired heuristic
- paradigm: evolution in nature, "survival of the fittest"
- actually it's only an algorithm, a randomised search heuristic (RSH)

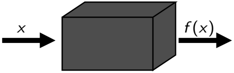


- Goal: optimisation
- Here: discrete search spaces, combinatorial optimisation, in particular pseudo-boolean functions

Optimise $f\colon \{0,1\}^n \rightarrow \mathbb{R}$

Why Do We Consider Randomised Search Heuristics?

- Not enough resources (time, money, knowledge) for a tailored algorithm

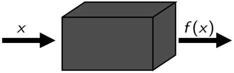
- Black Box Scenario 

rules out problem-specific algorithms

- We like the simplicity, robustness, ... of Randomised Search Heuristics

- "And they are surprisingly successful ..."

Why Do We Consider Randomised Search Heuristics?

- Not enough resources (time, money, knowledge) for a tailored algorithm
- Black Box Scenario 

  rules out problem-specific algorithms
- We like the simplicity, robustness, ... of Randomised Search Heuristics
- "And they are surprisingly successful ..."

Point of view

Do not only consider RSHs empirically. We need a solid theory to understand how (and when) they work.

What RSHs Do We Consider?

Theoretically considered RSHs

- (1+1) EA
- (1+λ) EA (offspring population)
- (μ+1) EA (parent population)
- (μ+1) GA (parent population and crossover)
- GIGA (crossover)
- SEMO, DEMO, FEMO, ... (multi-objective)
- Randomised Local Search (RLS)
- Metropolis Algorithm/Simulated Annealing (MA/SA)
- Ant Colony Optimisation (ACO)
- Particle Swarm Optimisation (PSO)
- ...

First of all: define the simple ones

The Most Basic RSHs

(1+1) EA, RLS, MA and SA for maximisation problems

(1+1) EA

1. Choose $x_0 \in \{0,1\}^n$ uniformly at random.
2. For $t := 0, \ldots, \infty$
   1. Create y by flipping each bit of x_t indep. with probab. $1/n$.
   2. If $f(y) \geq f(x_t)$ set $x_{t+1} := y$ else $x_{t+1} := x_t$.

The Most Basic RSHs

(1+1) EA, RLS, MA and SA for maximisation problems

RLS

1. Choose $x_0 \in \{0,1\}^n$ uniformly at random.
2. For $t := 0, \ldots, \infty$
   1. Create y by flipping one bit of x_t uniformly.
   2. If $f(y) \geq f(x_t)$ set $x_{t+1} := y$ else $x_{t+1} := x_t$.

(1+1) EA, RLS, MA and SA for maximisation problems

MA

1. Choose $x_0 \in \{0,1\}^n$ uniformly at random.
2. For $t := 0, \ldots, \infty$
   1. Create y by flipping one bit of x_t uniformly.
   2. If $f(y) \geq f(x_t)$ set $x_{t+1} := y$
      else $x_{t+1} := y$ with probability $e^{(f(x_t)-f(y))/T}$ anyway
      and $x_{t+1} := x_t$ otherwise.

T is fixed over all iterations.

(1+1) EA, RLS, MA and SA for maximisation problems

SA

1. Choose $x_0 \in \{0,1\}^n$ uniformly at random.
2. For $t := 0, \ldots, \infty$
   1. Create y by flipping one bit of x_t uniformly.
   2. If $f(y) \geq f(x_t)$ set $x_{t+1} := y$
      else $x_{t+1} := y$ with probability $e^{(f(x_t)-f(y))/T_t}$ anyway
      and $x_{t+1} := x_t$ otherwise.

T_t is dependent on t, typically decreasing

What Kind of Theory Are We Interested in?

- Not interesting here: convergence (often trivial), local progress, models of EAs (e. g., infinite populations), ...
- Treat RSHs as randomised algorithm!
- Analyse their "runtime" (computational complexity) on selected problems

What Kind of Theory Are We Interested in?

- Not interesting here: convergence (often trivial), local progress, models of EAs (e. g., infinite populations), ...
- Treat RSHs as randomised algorithm!
- Analyse their "runtime" (computational complexity) on selected problems

Definition

Let RSH A optimise f. Each f-evaluation is counted as a time step. The *runtime* $T_{A,f}$ of A is the random first point of time such that A has sampled an optimal search point.

- Often considered: expected runtime, distribution of $T_{A,f}$
- Asymptotical results w. r. t. n

How Do We Obtain Results?

We use (rarely in their pure form):

- Coupon Collector's Theorem
- Principle of Deferred Decisions
- Concentration inequalities:
  Markov, Chebyshev, Chernoff, Hoeffding, ... bounds
- Markov chain theory: waiting times, first hitting times
- Rapidly Mixing Markov Chains
- Random Walks: Gambler's Ruin, drift analysis (Wald's equation), martingale theory, electrical networks
- Random graphs (esp. random trees)
- Identifying typical events and failure events
- Potential functions and amortised analysis
- ...

How Do We Obtain Results?

We use (rarely in their pure form):

- Coupon Collector's Theorem
- Principle of Deferred Decisions
- Concentration inequalities:
  Markov, Chebyshev, Chernoff, Hoeffding, . . . bounds
- Markov chain theory: waiting times, first hitting times
- Rapidly Mixing Markov Chains
- Random Walks: Gambler's Ruin, drift analysis (Wald's equation), martingale theory, electrical networks
- Random graphs (esp. random trees)
- Identifying typical events and failure events
- Potential functions and amortised analysis
- . . .

Adapt tools from the analysis of randomised algorithms; understanding the stochastic process is often the hardest task.

Early Results

Analysis of RSHs already in the 1980s:

- Sasaki/Hajek (1988): SA and Maximum Matchings
- Sorkin (1991): SA vs. MA
- Jerrum (1992): SA and Cliques
- Jerrum/Sorkin (1993, 1998): SA/MA for Graph Bisection
- ...

These were high-quality results, however, limited to SA/MA (nothing about EAs) and hard to generalise.

Early Results

Analysis of RSHs already in the 1980s:

- Sasaki/Hajek (1988): SA and Maximum Matchings
- Sorkin (1991): SA vs. MA
- Jerrum (1992): SA and Cliques
- Jerrum/Sorkin (1993, 1998): SA/MA for Graph Bisection
- ...

These were high-quality results, however, limited to SA/MA (nothing about EAs) and hard to generalise.

Since the early 1990s

Systematic approach for the analysis of RSHs,
building up a completely new research area

This Tutorial

1. The origins: example functions and toy problems
   - A simple toy problem: OneMax for (1+1) EA
2. Combinatorial optimisation problems
   - (1+1) EA and Eulerian cycles
   - (1+1) EA and minimum spanning trees
   - (1+1) EA and maximum matchings
   - (1+1) EA and the partition problem
   - Multi-objective optimisation and the set cover problem
   - SA beats MA in combinatorial optimisation
   - ACO and minimum spanning trees
3. End
4. References

How the Systematic Research Began — Toy Problems

Simple example functions (test functions)

- $\text{OneMax}(x_1, \ldots, x_n) = x_1 + \cdots + x_n$
- $\text{LeadingOnes}(x_1, \ldots, x_n) = \sum_{i=1}^{n} \prod_{j=1}^{i} x_j$
- $\text{BinVal}(x_1, \ldots, x_n) = \sum_{i=1}^{n} 2^{n-i} x_i$
- polynomials of fixed degree

Goal: derive first runtime bounds and methods

How the Systematic Research Began — Toy Problems

Simple example functions (test functions)

- $\text{OneMax}(x_1, \ldots, x_n) = x_1 + \cdots + x_n$
- $\text{LeadingOnes}(x_1, \ldots, x_n) = \sum_{i=1}^{n} \prod_{j=1}^{i} x_j$
- $\text{BinVal}(x_1, \ldots, x_n) = \sum_{i=1}^{n} 2^{n-i} x_i$
- polynomials of fixed degree

Goal: derive first runtime bounds and methods

Artificially designed functions

- with sometimes really horrible definitions
- but for the first time these allow rigorous statements

Goal: prove benefits and harm of RSH components,
e. g., crossover, mutation strength, population size ...

This Tutorial

1. The origins: example functions and toy problems
   - A simple toy problem: OneMax for (1+1) EA
2. Combinatorial optimisation problems
   - (1+1) EA and Eulerian cycles
   - (1+1) EA and minimum spanning trees
   - (1+1) EA and maximum matchings
   - (1+1) EA and the partition problem
   - Multi-objective optimisation and the set cover problem
   - SA beats MA in combinatorial optimisation
   - ACO and minimum spanning trees
3. End
4. References

Example: OneMax

Theorem (e. g., Droste/Jansen/Wegener, 1998)

The expected runtime of the RLS, (1+1) EA, (μ+1) EA, (1+λ) EA on OneMax *is* $\Omega(n \log n)$.

Proof by modifications of Coupon Collector's Theorem.

Example: OneMax

Theorem (e. g., Droste/Jansen/Wegener, 1998)

The expected runtime of the RLS, (1+1) EA, (μ+1) EA, (1+λ) EA on $\textsc{OneMax}$ *is* $\Omega(n \log n)$.

Proof by modifications of Coupon Collector's Theorem.

Theorem (e. g., Mühlenbein, 1992)

The expected runtime of RLS and the (1+1) EA on $\textsc{OneMax}$ *is* $O(n \log n)$.

Holds also for population-based (μ+1) EA and
for (1+λ) EA with small populations.

Proof of the $O(n \log n)$ bound

- *Fitness levels:* $L_i := \{x \in \{0,1\}^n \mid \textsc{OneMax}(x) = i\}$

Proof of the $O(n \log n)$ bound

- *Fitness levels:* $L_i := \{x \in \{0,1\}^n \mid \textsc{OneMax}(x) = i\}$
- (1+1) EA never decreases its current fitness level.

Proof of the $O(n \log n)$ bound

- *Fitness levels:* $L_i := \{x \in \{0,1\}^n \mid \textsc{OneMax}(x) = i\}$
- (1+1) EA never decreases its current fitness level.
- From i to some higher-level set with prob. at least

$$\underbrace{\binom{n-i}{1}}_{\text{choose a 0-bit}} \cdot \underbrace{\left(\frac{1}{n}\right)}_{\text{flip this bit}} \cdot \underbrace{\left(1-\frac{1}{n}\right)^{n-1}}_{\text{keep the other bits}} \geq \frac{n-i}{en}$$

- Expected time to reach a higher-level set is at most $\frac{en}{n-i}$.
- Expected runtime is at most

$$\sum_{i=0}^{n-1} \frac{en}{n-i} = O(n \log n). \qquad \square$$

Later Results Using Toy Problems

- Find the theoretically optimal mutation strength ($1/n$ for OneMax!).
- optimal population size (often 1!)
- crossover vs. no crossover $\rightarrow$ Real Royal Road Functions
- multistarts vs. populations
- frequent restarts vs. long runs
- dynamic schedules
- ...

Later Results Using Toy Problems

- Find the theoretically optimal mutation strength ($1/n$ for OneMax!).
- optimal population size (often 1!)
- crossover vs. no crossover $\rightarrow$ Real Royal Road Functions
- multistarts vs. populations
- frequent restarts vs. long runs
- dynamic schedules
- ...

Further reading: Droste/Jansen/Wegener (2002), He/Yao (2002, 2003), Jansen (2002), Jansen/De Jong/Wegener (2005), Jansen/Wegener (2001, 2005), Storch/Wegener (2004), Witt (2006)

RSHs for Combinatorial Optimisation

- Analysis of runtime and approximation quality on well-known combinatorial optimisation problems, e. g.,
  - sorting problems (is this an optimisation problem?),
  - shortest path problems,
  - subsequence problems,
  - vertex cover,
  - Eulerian cycles,
  - minimum spanning trees,
  - maximum matchings,
  - partition problem,
  - set cover problem,
  - . . .

RSHs for Combinatorial Optimisation

- Analysis of runtime and approximation quality on well-known combinatorial optimisation problems, e. g.,
  - sorting problems (is this an optimisation problem?),
  - shortest path problems,
  - subsequence problems,
  - vertex cover,
  - Eulerian cycles,
  - minimum spanning trees,
  - maximum matchings,
  - partition problem,
  - set cover problem,
  - ...
- What we do not hope: to be better than the best problem-specific algorithms

RSHs for Combinatorial Optimisation

- Analysis of runtime and approximation quality on well-known combinatorial optimisation problems, e. g.,
  - sorting problems (is this an optimisation problem?),
  - shortest path problems,
  - subsequence problems,
  - vertex cover,
  - Eulerian cycles,
  - minimum spanning trees,
  - maximum matchings,
  - partition problem,
  - set cover problem,
  - ...
- What we do not hope: to be better than the best problem-specific algorithms
- In the following no fine-tuning of the results

Eulerian Cycle Problem

Given: Undirected connected Eulerian (degree of each vertex is even) graph $G = (V, E)$ with n vertices and m edges

Find: A Cycle (permutation of the edges) such that each edge is used exactly once.

Eulerian Cycle Problem

Given: Undirected connected Eulerian (degree of each vertex is even) graph $G = (V, E)$ with n vertices and m edges

Find: A Cycle (permutation of the edges) such that each edge is used exactly once.

Eulerian Cycle (Hierholzer)

1. Find a cycle C in G
2. Delete the edges of C from G
3. If G is not empty go to step 1.
4. Construct the Eulerian cycle from the cycles produced in Step 1.

Fitness Function

Representation: permutation of edges

Fitness function

Consider the edges of the permutation after another and build up a path p of length l.

$$path(\pi) := \text{ length of the path } p \text{ implied by } \pi$$

Example: $\pi = (\{2,3\}, \{1,2\}, \{1,5\}, \{3,4\}, \{4,5\}) \Longrightarrow |p| = 3$

The Algorithms

(1+1) EA

1. Choose $\pi \in S_m$ uniform at random.
2. Choose s according to a Poisson distribution with parameter $\lambda = 1$. Perform sequentially $s+1$ jump operations to produce π' from π.

(1+1) EA

1. Choose $\pi \in S_m$ uniform at random.
2. Choose s according to a Poisson distribution with parameter $\lambda = 1$. Perform sequentially $s + 1$ jump operations to produce π' from π.

   Example: jump(2,4) applied to
   ({2,3},{1,2},{3,4},{1,5},{4,5}) produces
   ({2,3},{3,4},{1,5},{1,2},{4,5})

(1+1) EA

1. Choose $\pi \in S_m$ uniform at random.
2. Choose s according to a Poisson distribution with parameter $\lambda = 1$. Perform sequentially $s+1$ jump operations to produce π' from π.
   Example: jump(2,4) applied to
   ({2,3},{1,2},{3,4},{1,5},{4,5}) produces
   ({2,3},{3,4},{1,5},{1,2},{4,5})
3. Replace π by π' if $path(\pi') \geq path(\pi)$.
4. Repeat Steps 2 and 3 forever.

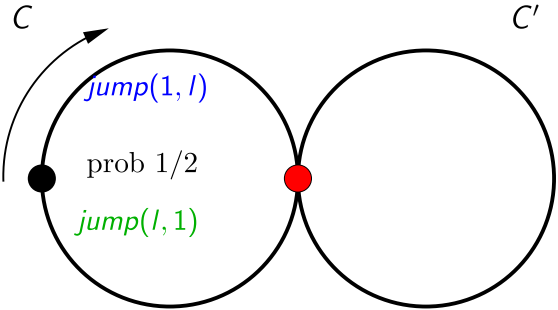
Upper Bound, (1+1) EA

Theorem (Neumann, 2007)

The expected time until (1+1) EA working on the fitness function path constructs an Eulerian cycle is bounded by $O(m^5)$.

Proof outline:

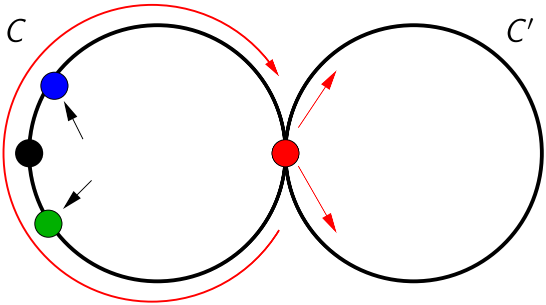
- p is not a cycle:
  1 improving jump $\Longrightarrow$ expected time for an improvement is $O(m^2)$
- p is a cycle:
  Show: Expected time for an improvement is bounded by $O(m^4)$
- $O(m)$ improvements $\Longrightarrow$ theorem



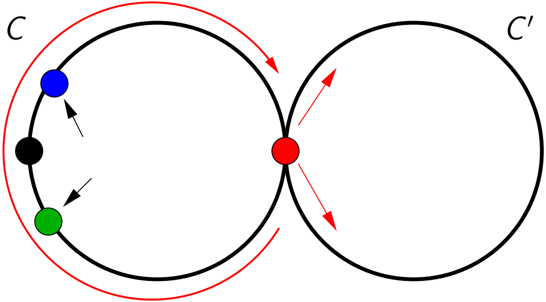
Typical run:

- k-step (accepted mutation with k-jumps that change p)
- Only 1-steps: $O(m^4)$ steps for an improvement
- No k-step, $k \geq 4$, in $O(m^4)$ steps with prob. $1 - o(1)$
- $O(1)$ 2- or 3-steps in $O(m^4)$ steps with prob. $1 - o(1)$

1-Steps



1-Steps



- time $O(m^2)$ to move black vertex
- black performs random walk
- length of cycle is at most m.
- fair random walk $\Longrightarrow O(m^2)$ movements are enough to reach red vertex
- expected time for an improvement $O(m^4)$

Further Results

- lower bound $\Omega(m^4)$

Further Results

- lower bound $\Omega(m^4)$
- restricted jumps (always jump to position 1)
  - no random walk, but directed walk
  - upper bound $O(m^3)$ (Doerr/Hebbinghaus/Neumann, 2007)

Further Results

- lower bound $\Omega(m^4)$
- restricted jumps (always jump to position 1)
  - no random walk, but directed walk
  - upper bound $O(m^3)$ (Doerr/Hebbinghaus/Neumann, 2007)
- use of more sophisticated representations and mutation operators:
  - $O(m^2 \log m)$ (Doerr/Klein/Storch, 2007)
  - $O(m \log m)$ (Doerr/Johannsen, 2007)

This Tutorial

Minimum Spanning Trees

Problem

Given: Undirected connected graph $G = (V, E)$ with n vertices and m edges with positive integer weights.
Find: Edge set $E' \subseteq E$ with minimal weight connecting all vertices.

Minimum Spanning Trees

Problem

Given: Undirected connected graph $G = (V, E)$ with n vertices and m edges with positive integer weights.
Find: Edge set $E' \subseteq E$ with minimal weight connecting all vertices.

Fitness function

Decrease number of connected components, find minimum spanning tree:

$$f(s) := (c(s), w(s)).$$

Minimization of f with respect to the lexicographic order.

Minimum Spanning Trees

Problem

Given: Undirected connected graph $G = (V, E)$ with n vertices and m edges with positive integer weights.
Find: Edge set $E' \subseteq E$ with minimal weight connecting all vertices.

Fitness function

Decrease number of connected components, find minimum spanning tree:

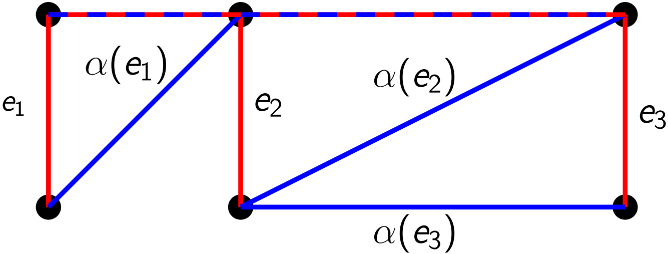
$$f(s) := (c(s), w(s)).$$

Minimization of f with respect to the lexicographic order.

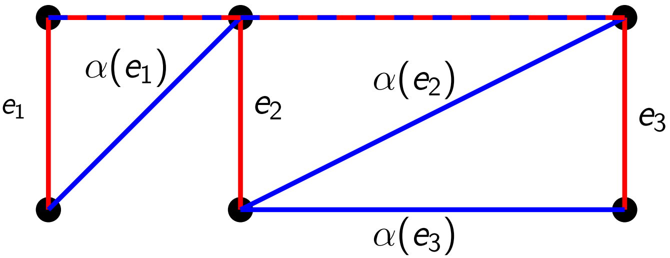
Connected graph

- Connected graph in expected time $O(m \log n)$ (fitness level arguments)

Bijection (Mayr/Plaxton, 1992)



Bijection (Mayr/Plaxton, 1992)



- $k := |E(T^*) \setminus E(T)|$
- Bijection $\alpha : E(T^*) \setminus E(T) \to E(T) \setminus E(T^*)$
- $\alpha(e_i)$ on the cycle of $E(T) \cup \{e_i\}$
- $w(e_i) \leq w(\alpha(e_i))$

$\Longrightarrow$ k accepted 2-bit flips that turn T into T^*

Upper Bound

Theorem (Neumann/Wegener, 2007)

The expected time until (1+1) EA constructs a minimum spanning tree is bounded by $O(m^2(\log n + \log w_{\max}))$.

Sketch of proof:

- $w(s)$ weight current solution s
- w_{opt} weight minimum spanning tree T^*

Upper Bound

Theorem (Neumann/Wegener, 2007)

The expected time until (1+1) EA constructs a minimum spanning tree is bounded by $O(m^2(\log n + \log w_{\max}))$.

Sketch of proof:

- $w(s)$ weight current solution s
- w_{opt} weight minimum spanning tree T^*
- set of $m+1$ operations to reach T^*
  - $m' = m - (n-1)$ 1-bit flips concerning non-T^* edges $\Longrightarrow$ spanning tree T
  - k 2-bit flips defined by bijection
  - $n-k$ non accepted 2-bit flips
- $\Longrightarrow$ average weight decrease $(w(s) - w_{opt})/(m+1)$

Upper Bound

- 1-step (larger total weight decrease of 1-bit flips)
- 2-step (larger total weight decrease of 2-bit flips)

- 1-step (larger total weight decrease of 1-bit flips)
- 2-step (larger total weight decrease of 2-bit flips)

Consider 2-steps:

- Expected weight decrease by a factor $1 - (1/(2n))$
- Probability $\Theta(n/m^2)$ for a good 2-bit flip
- Expected time until r 2-steps $O(rm^2/n)$

Upper Bound

- 1-step (larger total weight decrease of 1-bit flips)
- 2-step (larger total weight decrease of 2-bit flips)

Consider 2-steps:

- Expected weight decrease by a factor $1 - (1/(2n))$
- Probability $\Theta(n/m^2)$ for a good 2-bit flip
- Expected time until r 2-steps $O(rm^2/n)$

Consider 1-steps:

- Expected weight decrease by a factor $1 - (1/(2m'))$
- Probability $\Theta(m'/m)$ for a good 1-bit flip
- Expected time until r 1-steps $O(rm/m')$

Upper Bound

- 1-step (larger total weight decrease of 1-bit flips)
- 2-step (larger total weight decrease of 2-bit flips)

Consider 2-steps:

- Expected weight decrease by a factor $1 - (1/(2n))$
- Probability $\Theta(n/m^2)$ for a good 2-bit flip
- Expected time until r 2-steps $O(rm^2/n)$

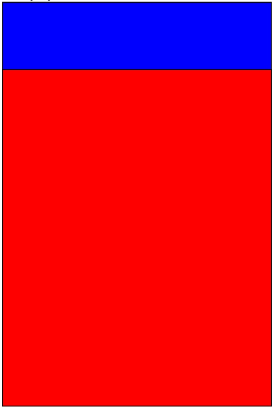
Consider 1-steps:

- Expected weight decrease by a factor $1 - (1/(2m'))$
- Probability $\Theta(m'/m)$ for a good 1-bit flip
- Expected time until r 1-steps $O(rm/m')$

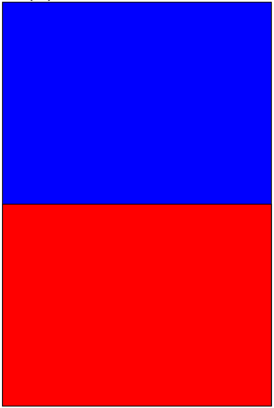
1-steps faster $\Longrightarrow$ show bound for 2-steps.

Expected Number of 2-Steps

$w(s) \leq D := m \cdot w_{max}$



$w(s) \leq D := m \cdot w_{max}$



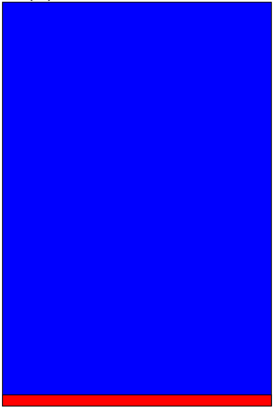
$(1 - \frac{1}{2n})(w(s) - w_{opt})$

$(1 - \frac{1}{2n})^N(w(s) - w_{opt})$

w_{opt}

Expected Number of 2-Steps

$w(s) \leq D := m \cdot w_{max}$



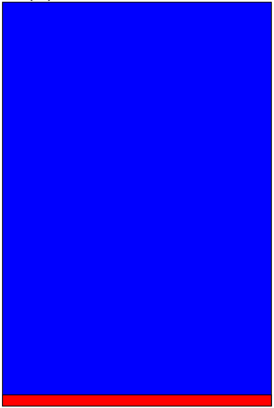
$(1 - \frac{1}{2n})(w(s) - w_{opt})$

$(1 - \frac{1}{2n})^N(w(s) - w_{opt})$

$N := \lceil 2 \cdot (\ln 2) \cdot n \cdot (\log D + 1) \rceil$

$\Longrightarrow (1 - \frac{1}{2n})^N(w(s) - w_{opt}) \leq 1/2$

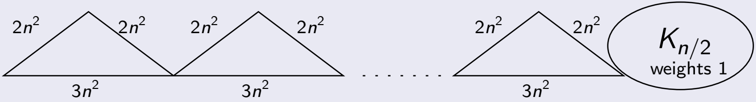
w_{opt}



- Expected number of 2-steps $2N = O(n(\log n + \log w_{max}))$ (Markov)
- Expected time $O(Nm^2/n) = O(m^2(\log n + \log w_{\max}))$.

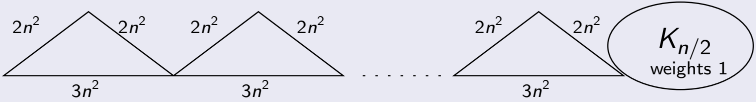
Further Results

Lower Bound $\Omega(n^4 \log n)$



Further Results

Lower Bound $\Omega(n^4 \log n)$



Related Results

- Experimental investigations (Briest et al., 2004)
- Biased mutation operators (Raidl/Koller/Julstrom, 2006)
- $O(mn^2)$ for a multi-objective approach (Neumann/Wegener, 2006)
- Approximations for multi-objective minimum spanning trees (Neumann, 2007)
- SA/MA/ACO and minimum spanning trees (Later!)

This Tutorial

1. The origins: example functions and toy problems
   - A simple toy problem: OneMax for (1+1) EA
2. Combinatorial optimisation problems
   - (1+1) EA and Eulerian cycles
   - (1+1) EA and minimum spanning trees
   - (1+1) EA and maximum matchings
   - (1+1) EA and the partition problem
   - Multi-objective optimisation and the set cover problem
   - SA beats MA in combinatorial optimisation
   - ACO and minimum spanning trees
3. End
4. References

$n+1$ nodes, n edges: bit string from $\{0,1\}^n$ selects edges

Fitness function: size of matching/negative for non-matchings

$n + 1$ nodes, n edges: bit string from $\{0, 1\}^n$ selects edges

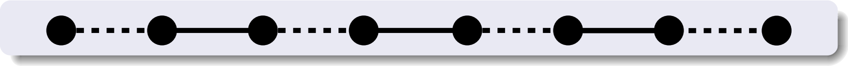
Fitness function: size of matching/negative for non-matchings

Theorem (Giel/Wegener, 2003)

The expected time until the (1+1) EA finds a maximum matching on a path of n edges is $O(n^4)$.

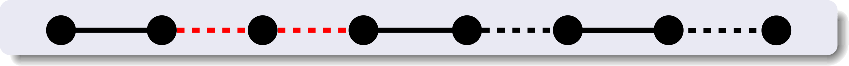
Proof idea:

- Consider a second-best matching.
- Is there a free edge? Flip one bit! $\rightarrow$ probability $\Theta(1/n)$.
- Else 2-bit flips $\rightarrow$ probability $\Theta(1/n^2)$.



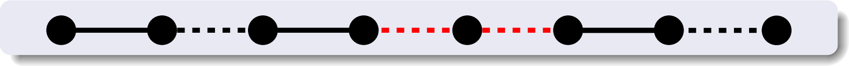
Proof idea:

- Consider a second-best matching.
- Is there a free edge? Flip one bit! $\rightarrow$ probability $\Theta(1/n)$.
- Else 2-bit flips $\rightarrow$ probability $\Theta(1/n^2)$.
- Shorten augmenting path



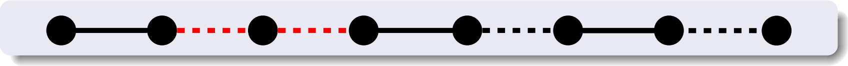
Proof idea:

- Consider a second-best matching.
- Is there a free edge? Flip one bit! $\rightarrow$ probability $\Theta(1/n)$.
- Else 2-bit flips $\rightarrow$ probability $\Theta(1/n^2)$.
- Shorten augmenting path



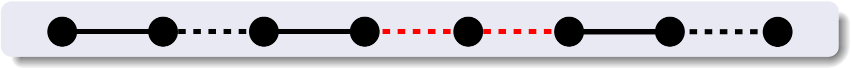
Proof idea:

- Consider a second-best matching.
- Is there a free edge? Flip one bit! $\rightarrow$ probability $\Theta(1/n)$.
- Else 2-bit flips $\rightarrow$ probability $\Theta(1/n^2)$.
- Shorten augmenting path



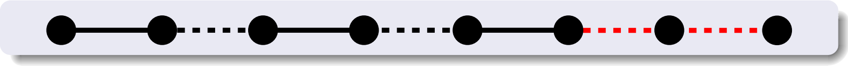
Proof idea:

- Consider a second-best matching.
- Is there a free edge? Flip one bit! $\rightarrow$ probability $\Theta(1/n)$.
- Else 2-bit flips $\rightarrow$ probability $\Theta(1/n^2)$.
- Shorten augmenting path



Proof idea:

- Consider a second-best matching.
- Is there a free edge? Flip one bit! $\rightarrow$ probability $\Theta(1/n)$.
- Else 2-bit flips $\rightarrow$ probability $\Theta(1/n^2)$.
- Shorten augmenting path
- Then flip the free edge!

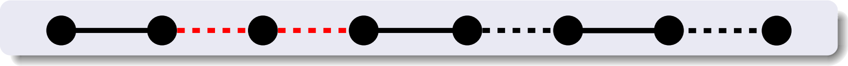


Proof idea:

- Consider a second-best matching.
- Is there a free edge? Flip one bit! $\rightarrow$ probability $\Theta(1/n)$.
- Else 2-bit flips $\rightarrow$ probability $\Theta(1/n^2)$.
- Shorten augmenting path
- Then flip the free edge!

Proof idea:

- Consider a second-best matching.
- Is there a free edge? Flip one bit! $\rightarrow$ probability $\Theta(1/n)$.
- Else 2-bit flips $\rightarrow$ probability $\Theta(1/n^2)$.
- Shorten augmenting path
- Then flip the free edge!



Proof idea:

- Consider a second-best matching.
- Is there a free edge? Flip one bit! $\rightarrow$ probability $\Theta(1/n)$.
- Else 2-bit flips $\rightarrow$ probability $\Theta(1/n^2)$.
- Shorten augmenting path
- Then flip the free edge!

Proof idea:

- Consider a second-best matching.
- Is there a free edge? Flip one bit! $\longrightarrow$ probability $\Theta(1/n)$.
- Else 2-bit flips $\longrightarrow$ probability $\Theta(1/n^2)$.
- Shorten augmenting path
- Then flip the free edge!

Proof idea:

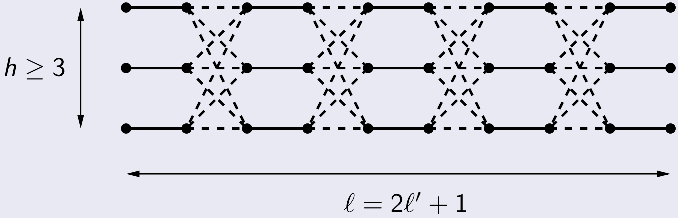
- Consider a second-best matching.
- Is there a free edge? Flip one bit! $\rightarrow$ probability $\Theta(1/n)$.
- Else 2-bit flips $\rightarrow$ probability $\Theta(1/n^2)$.
- Shorten augmenting path
- Then flip the free edge!
- (1+1) EA follows the concept of an augmenting path!

Proof idea:

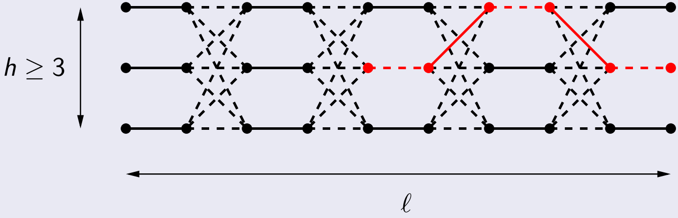
- Consider a second-best matching.
- Is there a free edge? Flip one bit! $\rightarrow$ probability $\Theta(1/n)$.
- Else 2-bit flips $\rightarrow$ probability $\Theta(1/n^2)$.
- Shorten augmenting path
- Then flip the free edge!
- (1+1) EA follows the concept of an augmenting path!

- Length changes according to a fair random walk (Gambler's Ruin Problem)
  $\rightarrow$ Expected runtime $O(n^2) \cdot O(n^2) = O(n^4)$.

Worst-case graph (Sasaki/Hajek, 1988)

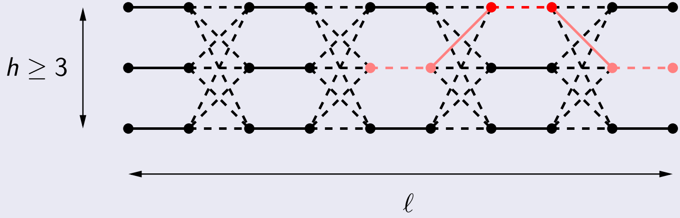


Worst-case graph (Sasaki/Hajek, 1988)



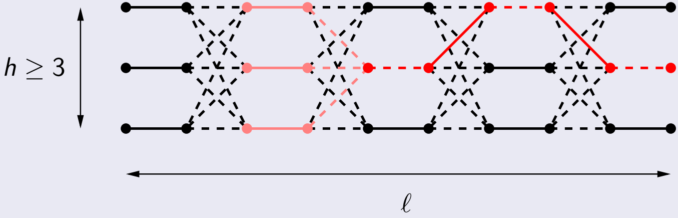
Augmenting path

Worst-case graph (Sasaki/Hajek, 1988)



Augmenting path can get shorter

Worst-case graph (Sasaki/Hajek, 1988)



Augmenting path can get shorter but is more likely to get longer.

Theorem

For $h \geq 3$, the (1+1) EA has exponential expected runtime $2^{\Omega(\ell)}$ on $G_{h,\ell}$.

Proof by drift analysis

Insight: do not hope for exact solutions but for approximations

Theorem (Giel/Wegener, 2003)

For $\varepsilon > 0$, the (1+1) EA finds a $(1+\varepsilon)$-approximation of a maximum matching in expected time $O(m^{2\lceil 1/\varepsilon \rceil})$ and is a polynomial-time randomised approximation scheme (PRAS).

Insight: do not hope for exact solutions but for approximations

Theorem (Giel/Wegener, 2003)

For $\varepsilon > 0$, the (1+1) EA finds a $(1+\varepsilon)$-approximation of a maximum matching in expected time $O(m^{2\lceil 1/\varepsilon\rceil})$ and is a polynomial-time randomised approximation scheme (PRAS).

Proof idea:

- Look into the analysis of the Hopcroft/Karp algorithm.
- Current solution worse than $(1+\varepsilon)$-approximate $\rightarrow$ many augmenting paths, in partic. a short one of length $\leq 2\lceil\varepsilon^{-1}\rceil$
- Wait for the (1+1) EA to optimise this short path.

A More General View

Minimum spanning trees and bipartite matching are special cases of matroid optimisation problems.

A More General View

Minimum spanning trees and bipartite matching are special cases of matroid optimisation problems.

Let E be a finite set and $\mathcal{F} \subseteq 2^E$. $M = (E, \mathcal{F})$ is a *matroid* if

(i) $\emptyset \in \mathcal{F}$,
(ii) $\forall X \subseteq Y \in \mathcal{F}\colon X \in \mathcal{F}$, and
(iii) $\forall X, Y \in \mathcal{F}, |X| > |Y|\colon \exists x \in X \setminus Y$ with $Y \cup \{x\} \in \mathcal{F}$.

Adding a function $w\colon E \to \mathbb{N}$ yields a weighted matroid.

A More General View

Minimum spanning trees and bipartite matching are special cases of matroid optimisation problems.

Let E be a finite set and $\mathcal{F} \subseteq 2^E$. $M = (E, \mathcal{F})$ is a *matroid* if

(i) $\emptyset \in \mathcal{F}$,

(ii) $\forall X \subseteq Y \in \mathcal{F}\colon X \in \mathcal{F}$, and

(iii) $\forall X, Y \in \mathcal{F}, |X| > |Y|\colon \exists x \in X \setminus Y$ with $Y \cup \{x\} \in \mathcal{F}$.

Adding a function $w\colon E \to \mathbb{N}$ yields a weighted matroid.

Exemplary Results (Reichel and Skutella, 2007)

The (1+1) EA and RLS solve the matroid optimisation problems

- min. weight basis exactly in time $O(|E|^2(\log|E| + \log w_{\max}))$.
- unweighted intersection up to $1 - \varepsilon$ in time $O(|E|^{2\lceil 1/\varepsilon \rceil})$.

A More General View

Minimum spanning trees and bipartite matching are special cases of matroid optimisation problems.

Let E be a finite set and $\mathcal{F} \subseteq 2^E$. $M = (E, \mathcal{F})$ is a *matroid* if

(i) $\emptyset \in \mathcal{F}$,

(ii) $\forall X \subseteq Y \in \mathcal{F}\colon X \in \mathcal{F}$, and

(iii) $\forall X, Y \in \mathcal{F}, |X| > |Y|\colon \exists x \in X \setminus Y$ with $Y \cup \{x\} \in \mathcal{F}$.

Adding a function $w\colon E \to \mathbb{N}$ yields a weighted matroid.

Exemplary Results (Reichel and Skutella, 2007)

The (1+1) EA and RLS solve the matroid optimisation problems

- min. weight basis exactly in time $O(|E|^2(\log |E| + \log w_{\max}))$.
- unweighted intersection up to $1 - \varepsilon$ in time $O(|E|^{2\lceil 1/\varepsilon \rceil})$.

Very abstract/general, a step towards a characterisation of polynomially solvable problems on which EAs are efficient

This Tutorial

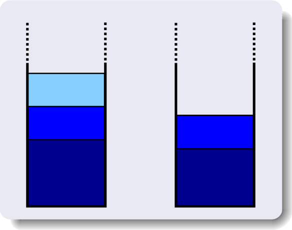
(1+1) EA and the Partition Problem

What about NP-hard problems? $\rightarrow$ Study approximation quality

What about NP-hard problems? → Study approximation quality

For $w_1, \ldots, w_n$, find $I \subseteq \{1, \ldots, n\}$ minimising

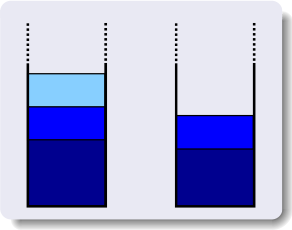
$$\max\left\{\sum_{i \in I} w_i, \sum_{i \notin I} w_i\right\}.$$



What about NP-hard problems? → Study approximation quality

For $w_1, \dots, w_n$, find $I \subseteq \{1, \dots, n\}$ minimising

$$\max \left\{ \sum_{i \in I} w_i, \sum_{i \notin I} w_i \right\}.$$



This is an "easy" NP-hard problem:

- not strongly NP-hard,
- FPTAS exist,
- ...

Coding: bit string $\{0,1\}^n$ characteristic vector of I

Fitness function: weight of fuller bin

Theorem (Witt, 2005)

On any instance for the partition problem, the (1+1) EA reaches a solution with approximation ratio $4/3$ *in expected time* $O(n^2)$.

Coding: bit string $\{0,1\}^n$ characteristic vector of I

Fitness function: weight of fuller bin

Theorem (Witt, 2005)

On any instance for the partition problem, the (1+1) EA reaches a solution with approximation ratio $4/3$ in expected time $O(n^2)$.

Theorem (Witt, 2005)

There is an instance such that the (1+1) EA needs with prob. $\Omega(1)$ at least $n^{\Omega(n)}$ steps to find a solution with a better ratio than $4/3 - \varepsilon$.

Proof ideas: study effect of local steps and local optima

Theorem (Witt, 2005)

On any instance, the (1+1) EA with prob. $\geq 2^{-c\lceil 1/\varepsilon\rceil \ln(1/\varepsilon)}$ finds a $(1+\varepsilon)$-approximation within $O(n \ln(1/\varepsilon))$ steps.

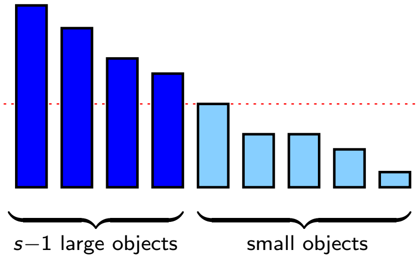
Theorem (Witt, 2005)

On any instance, the (1+1) EA with prob. $\geq 2^{-c\lceil 1/\varepsilon\rceil \ln(1/\varepsilon)}$ finds a $(1+\varepsilon)$-approximation within $O(n\ln(1/\varepsilon))$ steps.

- $2^{O(\lceil 1/\varepsilon\rceil \ln(1/\varepsilon))}$ parallel runs find a $(1+\varepsilon)$-approximation with prob. $\geq 3/4$ in $O(n\ln(1/\varepsilon))$ parallel steps.
- Parallel runs form a PRAS!

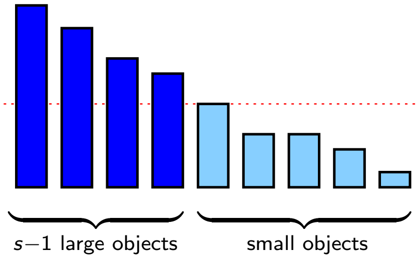
Set $s := \left\lceil \frac{2}{\varepsilon} \right\rceil$ and $w := \sum_{i=1}^{n} w_i$.

Assuming $w_1 \geq \cdots \geq w_n$, we have $w_i \leq \varepsilon \frac{w}{2}$ for $i \geq s$.



Set $s := \left\lceil \frac{2}{\varepsilon} \right\rceil$ and $w := \sum_{i=1}^{n} w_i$.

Assuming $w_1 \geq \cdots \geq w_n$, we have $w_i \leq \varepsilon \frac{w}{2}$ for $i \geq s$.



Analyse probability of distributing

- large objects in an optimal way,
- small objects greedily $\Rightarrow$ additive error $\leq \varepsilon w/2$,

This is the algorithmic idea by Graham (1969).

Models: each weight drawn independently at random, namely

1. uniformly from the interval $[0, 1]$,
2. exponentially distributed with parameter 1 (i. e., $\text{Prob}(X \geq t) = e^{-t}$ for $t \geq 0$).

Approximation ratio no longer meaningful, we investigate:
discrepancy = absolute difference between weights of bins.

Models: each weight drawn independently at random, namely

1. uniformly from the interval $[0,1]$,
2. exponentially distributed with parameter 1 (i. e., $\text{Prob}(X \geq t) = e^{-t}$ for $t \geq 0$).

Approximation ratio no longer meaningful, we investigate: discrepancy = absolute difference between weights of bins.

How close to discrepancy 0 do we come?

Deterministic, problem-specific heuristic LPT

Sort weights decreasingly,
put every object into currently emptier bin.

Analysis in both random models:

After LPT has been run, additive error is $O((\log n)/n)$ (Frenk/Rinnooy Kan, 1986).

Deterministic, problem-specific heuristic LPT

Sort weights decreasingly,
put every object into currently emptier bin.

Analysis in both random models:

After LPT has been run, additive error is $O((\log n)/n)$ (Frenk/Rinnooy Kan, 1986).

Can RLS or the (1+1) EA reach a discrepancy of $o(1)$?

Theorem (Witt, 2005)

In both models, the (1+1) EA reaches discrepancy $O((\log n)/n)$ after $O(n^{c+4}\log^2 n)$ steps with probability $1 - O(1/n^c)$.

Almost the same result as for LPT!

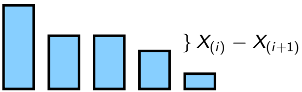
Theorem (Witt, 2005)

In both models, the (1+1) EA reaches discrepancy $O((\log n)/n)$ after $O(n^{c+4}\log^2 n)$ steps with probability $1 - O(1/n^c)$.

Almost the same result as for LPT!

Proof exploits order statistics:

W. h. p.
$X_{(i)} - X_{(i+1)} = O((\log n)/n)$
for $i = \Omega(n)$.

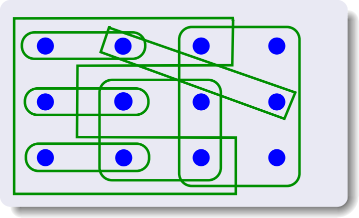


This Tutorial

1. The origins: example functions and toy problems
   - A simple toy problem: OneMax for (1+1) EA
2. Combinatorial optimisation problems
   - (1+1) EA and Eulerian cycles
   - (1+1) EA and minimum spanning trees
   - (1+1) EA and maximum matchings
   - (1+1) EA and the partition problem
   - Multi-objective optimisation and the set cover problem
   - SA beats MA in combinatorial optimisation
   - ACO and minimum spanning trees
3. End
4. References

The Set Cover Problem

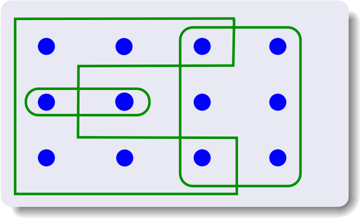
Another NP-hard problem



Given:

- ground set S,
- collection $C_1, \ldots, C_n$ of subsets with positive costs $c_1, \ldots, c_n$.

Another NP-hard problem



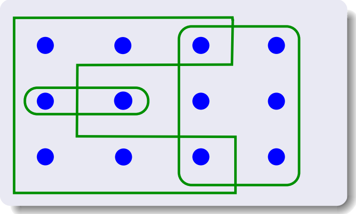
Given:

- ground set S,
- collection $C_1, \ldots, C_n$ of subsets with positive costs $c_1, \ldots, c_n$.

Goal: find a minimum-cost selection $C_{i_1}, \ldots, C_{i_k}$ such that $\bigcup_{j=1}^{k} C_{i_j} = S$.

The Set Cover Problem

Another NP-hard problem



Given:

- ground set S,
- collection $C_1, \ldots, C_n$ of subsets with positive costs $c_1, \ldots, c_n$.

Goal: find a minimum-cost selection $C_{i_1}, \ldots, C_{i_k}$ such that $\bigcup_{j=1}^{k} C_{i_j} = S$.

Traditional single-objective approach

Fitness = cost of selection of subsets, penalty for non-covers

Theorem

There is a Set Cover instance parameterised by $c > 0$ such that RLS and the (1+1) EA for any c need an infinite resp. exponential expected time to obtain a c-approximation.

Multi-objective Optimisation

Fitness $f\colon \{0,1\}^n \to \mathbb{R} \times \mathbb{R}$ has two objectives:

1. minimise the cost of the selection,
2. minimise the number of uncovered elements from S.

Multi-objective Optimisation

Fitness $f\colon \{0,1\}^n \rightarrow \mathbb{R} \times \mathbb{R}$ has two objectives:

1. minimise the cost of the selection,
2. minimise the number of uncovered elements from S.

Simple Evolutionary Multi-objective Optimiser (SEMO)

1. Choose $x \in \{0,1\}^n$ uniformly at random.
2. Determine $f(x)$.
3. $P \leftarrow \{x\}$.
4. Repeat
   - Choose $x \in P$ uniformly at random.
   - Create x' by flipping one randomly chosen bit of x.
   - Determine $f(x')$.
   - If x' is not dominated by any other search point in P, include x' into P and delete all other solutions $z \in P$ with $f(x') \preccurlyeq f(z)$ from P.

Achieving Almost Best-possible Approximations

Theorem (Friedrich, He, Hebbinghaus, Neumann, Witt, 2007)

For any instance of the Set Cover problem, SEMO finds an $(\ln|S| + 1)$*-approximate solution in expected time* $O(n|S|^2 + n|S|(\log n + \log c_{\max}))$*.*

Proof idea:

- Greedy procedure by cost-effectiveness: stepwise choose sets covering new elements at minimum average cost.

Achieving Almost Best-possible Approximations

Theorem (Friedrich, He, Hebbinghaus, Neumann, Witt, 2007)

For any instance of the Set Cover problem, SEMO finds an $(\ln|S|+1)$*-approximate solution in expected time* $O(n|S|^2 + n|S|(\log n + \log c_{\max}))$.

Proof idea:

- Greedy procedure by cost-effectiveness: stepwise choose sets covering new elements at minimum average cost.
- SEMO maintain covers with different numbers of uncovered elements.
- Potential function, value $k \Leftrightarrow$ SEMO covers k elements at cost $\leq \sum_{i=|S|-k+1}^{|S|} \frac{\mathrm{OPT}}{i}$.

Achieving Almost Best-possible Approximations

Theorem (Friedrich, He, Hebbinghaus, Neumann, Witt, 2007)

For any instance of the Set Cover problem, SEMO finds an $(\ln|S|+1)$-approximate solution in expected time $O(n|S|^2 + n|S|(\log n + \log c_{\max}))$.

Proof idea:

- Greedy procedure by cost-effectiveness: stepwise choose sets covering new elements at minimum average cost.
- SEMO maintain covers with different numbers of uncovered elements.
- Potential function, value $k \Leftrightarrow$ SEMO covers k elements at cost $\leq \sum_{i=|S|-k+1}^{|S|} \frac{\mathrm{OPT}}{i}$.
- Potential is increased by adding a most cost-effective set.
- Such step has probability $\Omega(1/(n|S|))$, at most $|S|$ increases to obtain approximation by factor $\sum_{i=1}^{|S|} 1/i \leq \ln|S|+1$.

Achieving Almost Best-possible Approximations

Theorem (Friedrich, He, Hebbinghaus, Neumann, Witt, 2007)

For any instance of the Set Cover problem, SEMO finds an $(\ln|S|+1)$-approximate solution in expected time $O(n|S|^2 + n|S|(\log n + \log c_{\max}))$.

Proof idea:

- Greedy procedure by cost-effectiveness: stepwise choose sets covering new elements at minimum average cost.
- SEMO maintain covers with different numbers of uncovered elements.
- Potential function, value $k \Leftrightarrow$ SEMO covers k elements at cost $\leq \sum_{i=|S|-k+1}^{|S|} \frac{\text{OPT}}{i}$.
- Potential is increased by adding a most cost-effective set.
- Such step has probability $\Omega(1/(n|S|))$, at most $|S|$ increases to obtain approximation by factor $\sum_{i=1}^{|S|} 1/i \leq \ln|S| + 1$.

It probably cannot be done better in polynomial time.

This Tutorial

1. The origins: example functions and toy problems
   - A simple toy problem: OneMax for (1+1) EA
2. Combinatorial optimisation problems
   - (1+1) EA and Eulerian cycles
   - (1+1) EA and minimum spanning trees
   - (1+1) EA and maximum matchings
   - (1+1) EA and the partition problem
   - Multi-objective optimisation and the set cover problem
   - SA beats MA in combinatorial optimisation
   - ACO and minimum spanning trees
3. End
4. References

Simulated Annealing Beats Metropolis in Combinatorial Optimisation

Jerrum/Sinclair (1996)

"It remains an outstanding open problem to exhibit a natural example in which simulated annealing with any non-trivial cooling schedule provably outperforms the Metropolis algorithm at a carefully chosen fixed value" of the temperature.

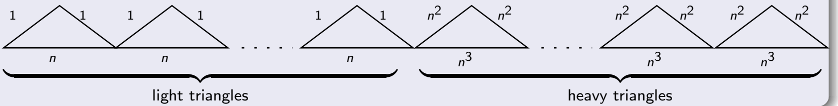
Simulated Annealing Beats Metropolis in Combinatorial Optimisation

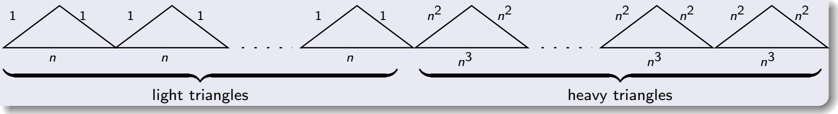
Jerrum/Sinclair (1996)

"It remains an outstanding open problem to exhibit a natural example in which simulated annealing with any non-trivial cooling schedule provably outperforms the Metropolis algorithm at a carefully chosen fixed value" of the temperature.

Solution (Wegener, 2005): MSTs are such an example.

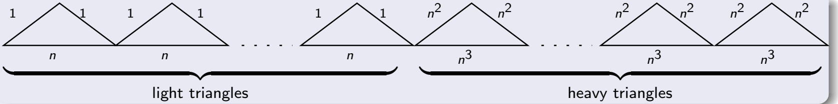
A bad instance for MA





Theorem (Wegener, 2005)

The MA with arbitrary temperature computes the MST for this instance only with probability $e^{-\Omega(n)}$ in polynomial time. SA with temperature $T_t := n^3(1 - \Theta(1/n))^t$ computes the MST in $O(n \log n)$ steps with probability $1 - O(1/poly(n))$.

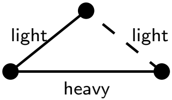


Theorem (Wegener, 2005)

The MA with arbitrary temperature computes the MST for this instance only with probability $e^{-\Omega(n)}$ in polynomial time. SA with temperature $T_t := n^3(1 - \Theta(1/n))^t$ computes the MST in $O(n \log n)$ steps with probability $1 - O(1/poly(n))$.

Proof idea: need different temperatures to optimise all triangles.

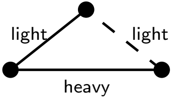
Concentrate on wrong triangles:
one heavy, one light edge chosen



Simulated Annealing Beats Metropolis in Combinatorial Optimisation

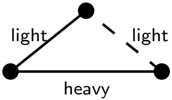
Proof Idea

Concentrate on wrong triangles:
one heavy, one light edge chosen



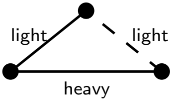
- Soon after initialization $\Omega(n)$ wrong triangles, both in heavy and light part of the graph
- To correct such triangle, light edge must be flipped in.

Concentrate on wrong triangles:
one heavy, one light edge chosen



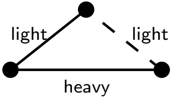
- Soon after initialization $\Omega(n)$ wrong triangles, both in heavy and light part of the graph
- To correct such triangle, light edge must be flipped in.
- Such flip leads to a worse spanning tree
  $\rightarrow$ need high temperature T^* to correct wrong heavy triangles.

Concentrate on wrong triangles:
one heavy, one light edge chosen



- Soon after initialization $\Omega(n)$ wrong triangles, both in heavy and light part of the graph
- To correct such triangle, light edge must be flipped in.
- Such flip leads to a worse spanning tree
  $\rightarrow$ need high temperature T^* to correct wrong heavy triangles.
- Light edges of heavy triangles still much heavier than heavy edges of light triangles $\rightarrow$ at temperature T^* almost random search on light triangles $\rightarrow$ many light triangles remain wrong.

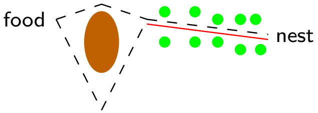
Concentrate on wrong triangles:
one heavy, one light edge chosen



- Soon after initialization $\Omega(n)$ wrong triangles, both in heavy and light part of the graph
- To correct such triangle, light edge must be flipped in.
- Such flip leads to a worse spanning tree $\rightarrow$ need high temperature T^* to correct wrong heavy triangles.
- Light edges of heavy triangles still much heavier than heavy edges of light triangles $\rightarrow$ at temperature T^* almost random search on light triangles $\rightarrow$ many light triangles remain wrong.
- SA first corrects heavy triangles at temperature T^*.
- After temperature has dropped, SA corrects light triangles, without destroying heavy ones.

This Tutorial

1. The origins: example functions and toy problems
   - A simple toy problem: OneMax for (1+1) EA
2. Combinatorial optimisation problems
   - (1+1) EA and Eulerian cycles
   - (1+1) EA and minimum spanning trees
   - (1+1) EA and maximum matchings
   - (1+1) EA and the partition problem
   - Multi-objective optimisation and the set cover problem
   - SA beats MA in combinatorial optimisation
   - ACO and minimum spanning trees
3. End
4. References

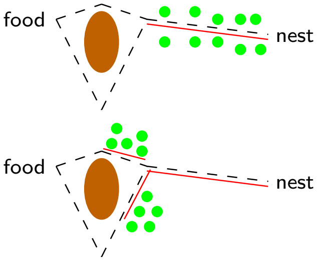


Ant colonies in nature

- find shortest paths
  in an unknown environment
- using communication via
  pheromone trails
- show adaptive behaviour

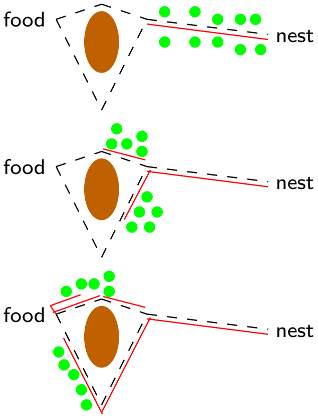
Ant colonies in nature

- find shortest paths
  in an unknown environment
- using communication via
  pheromone trails
- show adaptive behaviour



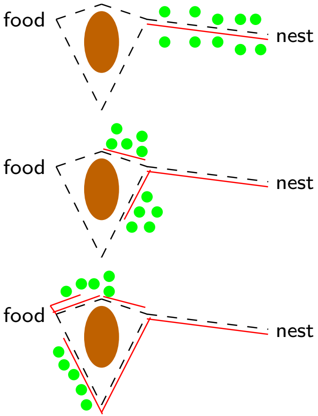
Ant colonies in nature

- find shortest paths in an unknown environment
- using communication via pheromone trails
- show adaptive behaviour



Ant colonies in nature

- find shortest paths in an unknown environment
- using communication via pheromone trails
- show adaptive behaviour



Ant Colony Optimisation (ACO) is yet another biologically inspired search heuristic.

Applications: combinatorial optimisation problems, e. g., TSP

Broder's Algorithm

Problem: Minimum Spanning Trees

Broder's Algorithm

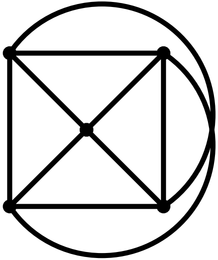
Problem: Minimum Spanning Trees

Consider the input graph itself as construction graph.

Problem: Minimum Spanning Trees

Consider the input graph itself as construction graph.

Spanning tree can be chosen uniformly at random using random walk algorithms (e. g. Broder, 1989).

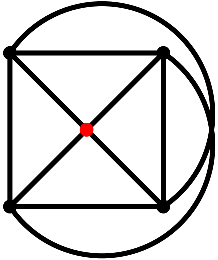


Reward chosen edges ⇒ next solution will be similar to constructed one
But: local improvements are possible

Problem: Minimum Spanning Trees

Consider the input graph itself as construction graph.

Spanning tree can be chosen uniformly at random using random walk algorithms (e. g. Broder, 1989).



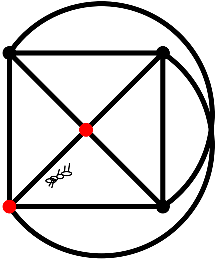
Reward chosen edges ⇒ next solution will be similar to constructed one
But: local improvements are possible

Broder's Algorithm

Problem: Minimum Spanning Trees

Consider the input graph itself as construction graph.

Spanning tree can be chosen uniformly at random using random walk algorithms (e. g. Broder, 1989).

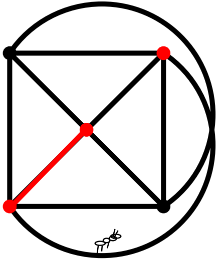


Reward chosen edges ⇒ next solution will be similar to constructed one
But: local improvements are possible

Problem: Minimum Spanning Trees

Consider the input graph itself as construction graph.

Spanning tree can be chosen uniformly at random using random walk algorithms (e. g. Broder, 1989).



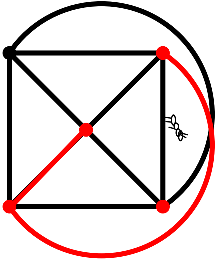
Reward chosen edges ⇒ next solution will be similar to constructed one
But: local improvements are possible

Broder's Algorithm

Problem: Minimum Spanning Trees

Consider the input graph itself as construction graph.

Spanning tree can be chosen uniformly at random using random walk algorithms (e. g. Broder, 1989).



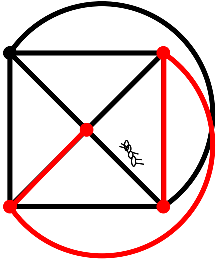
Reward chosen edges ⇒ next solution will be similar to constructed one
But: local improvements are possible

Broder's Algorithm

Problem: Minimum Spanning Trees

Consider the input graph itself as construction graph.

Spanning tree can be chosen uniformly at random using random walk algorithms (e. g. Broder, 1989).

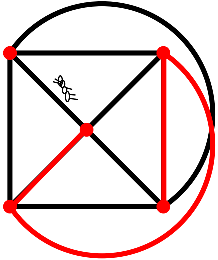


Reward chosen edges $\Rightarrow$ next solution will be similar to constructed one
But: local improvements are possible

Problem: Minimum Spanning Trees

Consider the input graph itself as construction graph.

Spanning tree can be chosen uniformly at random using random walk algorithms (e. g. Broder, 1989).

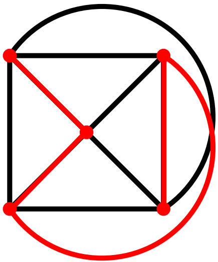


Reward chosen edges ⇒ next solution will be similar to constructed one
But: local improvements are possible

Problem: Minimum Spanning Trees

Consider the input graph itself as construction graph.

Spanning tree can be chosen uniformly at random using random walk algorithms (e. g. Broder, 1989).



Reward chosen edges ⇒ next solution will be similar to constructed one

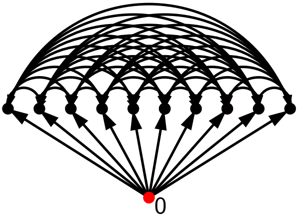
But: local improvements are possible

Component-based Construction Graph

- Vertices correspond to edges of the input graph
- Construction graph $C(G) = (N, A)$ satisfies $N = \{0, \ldots, m\}$ (start vertex 0) and $A = \{(i, j) \mid 0 \leq i \leq m, 1 \leq j \leq m, i \neq j\}$.

Component-based Construction Graph

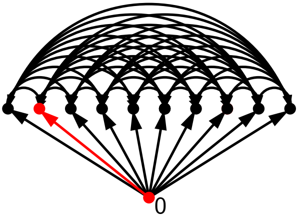
- Vertices correspond to edges of the input graph
- Construction graph $C(G) = (N, A)$ satisfies $N = \{0, \ldots, m\}$ (start vertex 0) and $A = \{(i, j) \mid 0 \leq i \leq m, 1 \leq j \leq m, i \neq j\}$.



For a given path $v_1, \ldots, v_k$ select the next edge from its neighborhood $N(v_1, \ldots, v_k) :=$ $(E \setminus \{v_1, \ldots, v_k\}) \setminus \{e \in E \mid (V, \{v_1, \ldots, v_k, e\}) \text{ contains a cycle}\}$

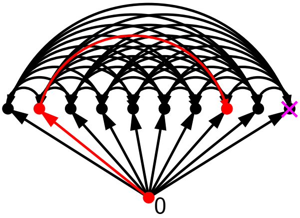
(problem-specific aspect of ACO).

- Vertices correspond to edges of the input graph
- Construction graph $C(G) = (N, A)$ satisfies $N = \{0, \ldots, m\}$ (start vertex 0) and $A = \{(i,j) \mid 0 \leq i \leq m, 1 \leq j \leq m, i \neq j\}$.



For a given path $v_1, \ldots, v_k$ select the next edge from its neighborhood
$N(v_1, \ldots, v_k) :=$
$(E \setminus \{v_1, \ldots, v_k\}) \setminus \{e \in E \mid$
$(V, \{v_1, \ldots, v_k, e\})$ contains a cycle$\}$
(problem-specific aspect of ACO).

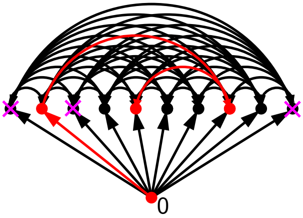
- Vertices correspond to edges of the input graph
- Construction graph $C(G) = (N, A)$ satisfies $N = \{0, \ldots, m\}$ (start vertex 0) and $A = \{(i, j) \mid 0 \leq i \leq m, 1 \leq j \leq m, i \neq j\}$.



For a given path $v_1, \ldots, v_k$ select the next edge from its neighborhood $N(v_1, \ldots, v_k) :=$ $(E \setminus \{v_1, \ldots, v_k\}) \setminus \{e \in E \mid (V, \{v_1, \ldots, v_k, e\})$ contains a cycle$\}$

(problem-specific aspect of ACO).

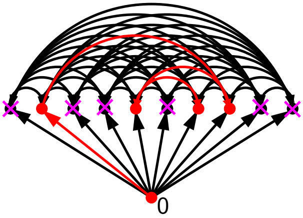
- Vertices correspond to edges of the input graph
- Construction graph $C(G) = (N, A)$ satisfies $N = \{0, \ldots, m\}$ (start vertex 0) and $A = \{(i, j) \mid 0 \le i \le m, 1 \le j \le m, i \ne j\}$.



For a given path $v_1, \ldots, v_k$ select the next edge from its neighborhood $N(v_1, \ldots, v_k) :=$ $(E \setminus \{v_1, \ldots, v_k\}) \setminus \{e \in E \mid$ $(V, \{v_1, \ldots, v_k, e\})$ contains a cycle$\}$

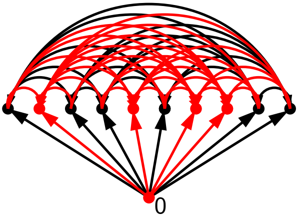
(problem-specific aspect of ACO).

- Vertices correspond to edges of the input graph
- Construction graph $C(G) = (N, A)$ satisfies $N = \{0, \ldots, m\}$ (start vertex 0) and $A = \{(i, j) \mid 0 \leq i \leq m, 1 \leq j \leq m, i \neq j\}$.



Reward: all edges, that point to visited vertices
(neglect order of chosen edges)

- Vertices correspond to edges of the input graph
- Construction graph $C(G) = (N, A)$ satisfies $N = \{0, \ldots, m\}$ (start vertex 0) and $A = \{(i, j) \mid 0 \leq i \leq m, 1 \leq j \leq m, i \neq j\}$.



1-ANT:

- two pheromone values
- value h: if edge has been rewarded
- value ℓ: otherwise
- heuristic information η, $\eta(e) = \frac{1}{w(e)}$ (used before for TSP)

1-ANT:

- two pheromone values
- value h: if edge has been rewarded
- value ℓ: otherwise
- heuristic information η, $\eta(e) = \frac{1}{w(e)}$ (used before for TSP)
- Let v_k the current vertex and N_{v_k} be its neighborhood.
- Prob(to choose neighbor y of v_k) $= \frac{[\tau_{(v_k,y)}]^\alpha \cdot [\eta_{(v_k,y)}]^\beta}{\sum_{y \in N(v_k)} [\tau_{(v_k,y)}]^\alpha \cdot [\eta_{(v_k,y)}]^\beta}$ with $\alpha, \beta \geq 0$.
- Consider special cases where either $\beta = 0$ or $\alpha = 0$.

Results for Pheromone Updates

Case $\alpha = 1$, $\beta = 0$: proportional influence of pheromone values

Case $\alpha = 1$, $\beta = 0$: proportional influence of pheromone values

Theorem (Broder-based construction graph)

Choosing $h/\ell = n^3$, the expected time until the 1-ANT with the Broder-based construction graph has found an MST is $O(n^6(\log n + \log w_{max}))$.

Results for Pheromone Updates

Case $\alpha = 1$, $\beta = 0$: proportional influence of pheromone values

Theorem (Broder-based construction graph)

Choosing $h/\ell = n^3$*, the expected time until the 1-ANT with the Broder-based construction graph has found an MST is* $O(n^6(\log n + \log w_{max}))$.

Theorem (Component-based construction graph)

Choosing $h/\ell = (m - n + 1)\log n$*, the expected time until the 1-ANT with the component-based construction graph has found an MST is* $O(mn(\log n + \log w_{max}))$.

Results for Pheromone Updates

Case $\alpha = 1$, $\beta = 0$: proportional influence of pheromone values

Theorem (Broder-based construction graph)

Choosing $h/\ell = n^3$, the expected time until the 1-ANT with the Broder-based construction graph has found an MST is $O(n^6(\log n + \log w_{max}))$.

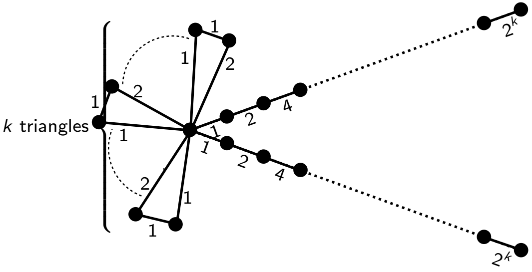
Theorem (Component-based construction graph)

Choosing $h/\ell = (m - n + 1)\log n$, the expected time until the 1-ANT with the component-based construction graph has found an MST is $O(mn(\log n + \log w_{max}))$.

Better than (1+1) EA!

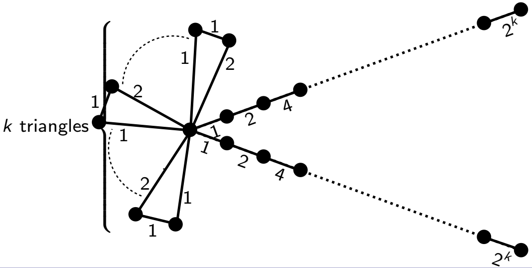
Example graph G^* with $n = 4k + 1$ vertices.

- k triangles of weight profile $(1, 1, 2)$
- two paths of length k with exponentially increasing weights.



Example graph G^* with $n = 4k + 1$ vertices.

- k triangles of weight profile $(1, 1, 2)$
- two paths of length k with exponentially increasing weights.



Theorem (Broder-based construction graph)

Let $\alpha = 0$ and β be arbitrary, then the probability that the 1-ANT using the Broder construction procedure does not find an MST in polynomial time with probability $1 - 2^{-\Omega(n)}$.

Component-based Construction Graph/Heuristic Information

Theorem (Component-based construction graph)

Choosing $\alpha = 0$ and $\beta \geq 6w_{\mathrm{max}} \log n$, the expected time of the 1-ANT with the component-based construction graph to find an MST is constant.

Component-based Construction Graph/Heuristic Information

Theorem (Component-based construction graph)

Choosing $\alpha = 0$ and $\beta \geq 6w_{\max} \log n$, the expected time of the 1-ANT with the component-based construction graph to find an MST is constant.

Proof Idea

- Choose edges as Kruskal's algorithm.
- Calculation shows: probability of choosing a lightest edge is at least $(1 - 1/n)$.
- $n - 1$ steps $\Longrightarrow$ probability for an MST is $\Omega(1)$.

Summary and Conclusions

- Analysis of RSHs in combinatorial optimisation
- Starting from toy problems to real problems
- Surprising results
- Interesting techniques
- Can analyse even new approaches

Summary and Conclusions

- Analysis of RSHs in combinatorial optimisation
- Starting from toy problems to real problems
- Surprising results
- Interesting techniques
- Can analyse even new approaches

$\longrightarrow$ The analysis of RSHs is an exciting research direction.

Thank you!

Selected Literature I

P. Briest and D. Brockhoff and B. Degener and M. Englert and C. Gunia and O. Heering and T. Jansen and M. Leifhelm and K. Plociennik and H. Röglin and A. Schweer and D. Sudholt and S. Tannenbaum and I. Wegener: (2004):
Experimental supplements to the theoretical analysis of EAs on problems from combinatorial optimization.
Proc. of PPSN 2004, 21–30, Springer

A. Z. Broder (1989):
Generating random spanning trees.
Proc. of FOCS 1989, 442–447

B. Doerr and N. Hebbinghaus and F. Neumann (2007):
Speeding up evolutionary algorithms through unsymmetric mutation operators.
Evolutionary Computation, 15(4): 401–410

B. Doerr and D. Johanssen (2007):
Adjacency list matchings - an ideal genotype for cycle covers.
Proc. of GECCO 2007, 1203–1210, ACM Press

B. Doerr and D. Johanssen (2007):
Refined runtime analysis of a basic ant colony optimization algorithm.
Proc. of CEC 2007, 501–507, IEEE Press

B. Doerr and C. Klein and T. Storch (2007):
Faster evolutionary algorithms by superior graph representation.
Proc. of FOCI 2007, 245–250, IEEE Press

T. Friedrich and J. He and N. Hebbinghaus and F. Neumann and C. Witt (2007):
Approximating Covering Problems by Randomized Search Heuristics Using Multi-Objective Models.
Proc. of GECCO 2007, 797–804, ACM Press

B. Doerr and F. Neumann and D. Sudholt and C. Witt (2007):
On the runtime analysis of the 1-ANT ACO algorithm.
Proc. of GECCO 2007, 33–40, ACM Press

S. Droste and T. Jansen and I. Wegener (1998):
A rigorous complexity analysis of the (1+1) evolutionary algorithm for separable functions with boolean inputs.
Evolutionary Computation, 6(2):185–196

S. Droste, T. Jansen and I. Wegener (2002):
On the analysis of the (1+1) evolutionary algorithm.
Theoretical Computer Science, 276:51–81, 2002

O. Giel and I. Wegener (2003):
Evolutionary algorithms and the maximum matching problem.
Proc. of STACS '03, LNCS 2607, 415–426, Springer

R. L. Graham (1969):
Bounds on multiprocessing timing anomalies.
SIAM Journal of Applied Mathematics, 17(2): 416–429

J. He and X. Yao (2001):
Drift analysis and average time complexity of evolutionary algorithms.
Artificial Intelligence, 127(1), 57–85

J. He and X. Yao (2002):
Erratum to: Drift analysis and average time complexity of evolutionary algorithms.
Artificial Intelligence, 140(1), 245–248

J. He and X. Yao (2003):
Towards an analytic framework for analysing the computation time of evolutionary algorithms.
Artificial Intelligence, 145(1–2), 59–97

T. Jansen (2002):
On the analysis of dynamic restart strategies for evolutionary algorithms.
Proc. of PPSN VIII, LNCS 2439, 33–43, Springer

T. Jansen, K. A. De Jong and I. Wegener (2005):
On the choice of the offspring population size in evolutionary algorithms.
Evolutionary Computation, 13(4): 413–440, 2005

T. Jansen and I. Wegener (2001):
On the utility of populations.
Proc. of GECCO 2001, 1034–1041, Morgan Kaufmann

T. Jansen and I. Wegener (2002):
The analysis of evolutionary algorithms – a proof that crossover really can help.
Algorithmica, 34:47–66

T. Jansen and I. Wegener (2005):
Real royal road functions – where crossover provably is essential.
Discrete Applied Mathematics, 149:111-125

T. Jansen and I. Wegener (2006):
On the analysis of a dynamic evolutionary algorithm.
Journal of Discrete Algorithms, 4(1):181-199

Selected Literature IV

M. Jerrum (1992):
Large cliques elude the Metropolis process.
Random Structures and Algorithms, 3(4):347–360

M. Jerrum and G. B. Sorkin (1998):
The Metropolis algorithm for graph bisection.
Discrete Applied Mathematics, 82(1–3):155–175. Preliminary version in FOCS 1993

E. W. Mayer and C. W. Plaxton (1992):
On the spanning trees of weighted graphs.
Combinatorica, 12(4), 433–447

H. Mühlenbein (1992):
How genetic algorithms really work: mutation and hill-climbing.
Proc. of PPSN II, 15–26, North Holland

F. Neumann (2007):
Expected runtimes of a simple evolutionary algorithm for the multi-objective minimum spanning tree problem.
European Journal of Operational Research, 181(3): 1620–1629

F. Neumann (2007):
Expected runtimes of evolutionary algorithms for the Eulerian cycle problem.
Computers and Operations Research, 35(9): 2750–2759

F. Neumann and I. Wegener (2006):
Minimum spanning trees made easier via multi-objective optimization.
Natural Computing, 5(3): 305–319

Selected Literature V

F. Neumann and I. Wegener (2007):
Randomized local search, evolutionary algorithms, and the minimum spanning tree problem.
Theoretical Computer Science, 378(1): 32–40

F. Neumann and C. Witt (2006):
Runtime analysis of a simple ant colony optimization algorithm.
Proc. of ISAAC 2006, 618–627, Springer, extended version to appear in Algorithmica (2008)

F. Neumann and C. Witt (2008):
Ant colony optimization and the minimum spanning tree problem.
Proc. of LION II (to appear).
Preliminary version in Electronic Colloquium on Computational Complexity (ECCC), Report No. 143

G. R. Raidl and G. Koller and B. A. Julstrom: (2006):
Biased mutation operators for subgraph-selection problems.
IEEE Trans. Evolutionary Computation, 10(2): 145–156

J. Reichel and M. Skutella (2007):
Evolutionary algorithms and matroid optimization problems.
Proc. of GECCO 2007, 947–954, ACM Press

G. H. Sasaki and B. Hajek (1988):
The time complexity of maximum matching by simulated annealing.
Journal of the ACM, 35(2): 387–403

G. B. Sorkin (1991):
Efficient simulated annealing on fractal energy landscapes.
Algorithmica, 6(3): 367–418

Selected Literature VI

T. Storch and I. Wegener (2004):
Real royal road functions for constant population size.
Theoretical Computer Science, 320(1): 123–134.

I. Wegener (2005):
Simulated annealing beats Metropolis in combinatorial optimization.
Proc. of ICALP 2005, LNCS 3580, 589–601, Springer

C. Witt (2005):
Worst-case and average-case approximations by simple randomized search heuristics.
Proc. of STACS 2005, LNCS 3404, 44-56, Springer

C. Witt (2006):
Runtime analysis of the $(\mu+1)$ EA on simple pseudo-boolean functions.
Evolutionary Computation, 14(1), 65–86