Theory of Randomised Search Heuristics in Combinatorial Optimisation

An Algorithmic Point of View

Carsten Witt

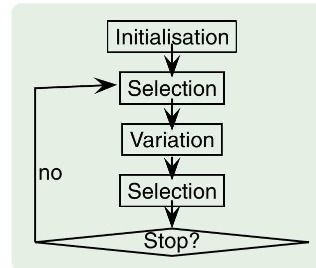
Fakultät für Informatik, LS 2
Technische Universität Dortmund
Germany

Tutorial at GECCO 2008
13 July 2008

What Are Randomised Search Heuristics (RSHs)?

Most famous example: Evolutionary Algorithms (EAs)

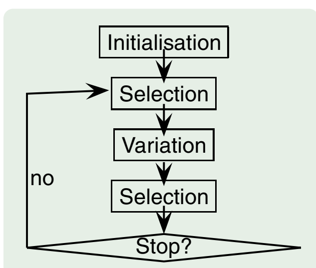
- a bio-inspired heuristic
- paradigm: evolution in nature, "survival of the fittest"



What Are Randomised Search Heuristics (RSHs)?

Most famous example: Evolutionary Algorithms (EAs)

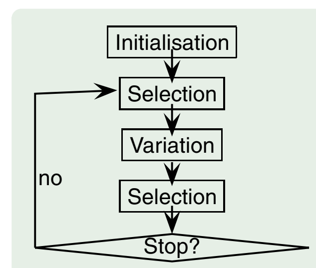
- a bio-inspired heuristic
- paradigm: evolution in nature, "survival of the fittest"
- actually it's only an algorithm, a randomised search heuristic



What Are Randomised Search Heuristics (RSHs)?

Most famous example: Evolutionary Algorithms (EAs)

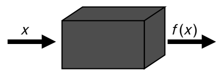
- a bio-inspired heuristic
- paradigm: evolution in nature, "survival of the fittest"
- actually it's only an algorithm, a randomised search heuristic



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

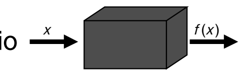
Optimise $f\colon \{0,1\}^n \to \mathbb{R}$

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 ...”

My 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 (multi-objective)
- Randomised Local Search (RLS)
- Metropolis Algorithm/Simulated Annealing (MA/SA)
- Ant Colony Optimisation (ACO)
- ...

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 $t := 1, \ldots, \infty$
   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 $t := 1, \dots, \infty$
   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 $x_{t+1} := x_t$.

The Most Basic RSHs

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

MA

1. Choose $x_0 \in \{0,1\}^n$ uniformly at random.
2. For $t := 1, \dots, \infty$
   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.

The Most Basic RSHs

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

SA

1. Choose $x_0 \in \{0,1\}^n$ uniformly at random.
2. For $t := 1, \dots, \infty$
   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" 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" 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

How the Systematic Research Began — Toy Problems

Simple example functions (test functions)

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

Goal: derive first runtime bounds and methods

How the Systematic Research Began — Toy Problems

Simple example functions (test functions)

- $\text{OneMax}(x_1, \dots, x_n) = x_1 + \dots + x_n$
- $\text{LeadingOnes}(x_1, \dots, x_n) = \sum_{i=1}^{n} \prod_{j=1}^{i} x_j$
- $\text{BinVal}(x_1, \dots, x_n) = \sum_{i=1}^{n} 2^{n-i} x_i$
- 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
   - An advanced example: population size for the (μ+1) EA
2. Combinatorial optimisation problems
   - (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 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 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:* $L_i := \{x \in \{0,1\}^n \mid |x|_1 = i\}$

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

- *Fitness levels:* $L_i := \{x \in \{0,1\}^n \mid |x|_1 = i\}$
- (1+1) EA never decreases its current fitness level.

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

- *Fitness levels:* $L_i := \{x \in \{0,1\}^n \mid |x|_1 = i\}$
- (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 Example Functions

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

Later Results Using Example Functions

- Find the theoretically optimal mutation strength ($1/n$ for OneMax!).
- optimal population size (often 1!)
- crossover vs. no crossover → 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)

This Tutorial

1. The origins: example functions and toy problems
   - A simple toy problem: OneMax for (1+1) EA
   - An advanced example: population size for the (μ+1) EA
2. Combinatorial optimisation problems
   - (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

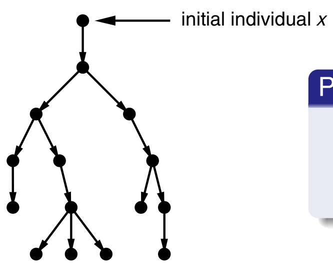
An Advanced Example: (μ+1) EA
Definition

(μ+1) EA

Convention: multisets

1. Choose $P_0 := \{x_1, \ldots, x_\mu \in \{0,1\}^n\}$ uniformly at random.
2. For $t := 1, \ldots, \infty$
   1. Choose x from P_t uniformly at random.
   2. Create y by flipping each bit of x indep. with probab. $1/n$.
   3. Set $P^* := P_t \cup \{y\}$.
   4. Choose x with lowest f-value in P^* uniformly.
   5. Set $P_{t+1} := P^* \setminus \{x\}$.

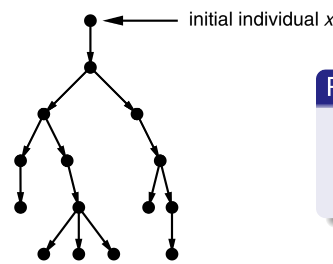
Advanced Example: (μ+1) EA and Family Trees
Properties of Trees



Properties

- nodes = descendants of x
- new node after each mutation of a descendant

Advanced Example: (μ+1) EA and Family Trees
Properties of Trees



Properties

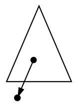
- nodes = descendants of x
- new node after each mutation of a descendant

- Interesting: depth of the tree since low depth $\longrightarrow$ few progress
- What stochastic process creates the tree?

The Process Behind Family Trees

Sequence of trees T_t such that

- at time 0, there is only the root,
- at time t, either nothing happens ($T_{t+1} = T_t$), or



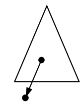
node from T_t is chosen and new leaf appended $\rightarrow T_{t+1}$.

Crucial: each node chosen with prob. at most $1/\mu$.

The Process Behind Family Trees

Sequence of trees T_t such that

- at time 0, there is only the root,
- at time t, either nothing happens ($T_{t+1} = T_t$), or



node from T_t is chosen and new leaf appended $\rightarrow T_{t+1}$.

Crucial: each node chosen with prob. at most $1/\mu$.

Technical Lemma (Witt, 2006)

Depth of tree at time t: at most $\frac{3t}{\mu}$ with prob. $1 - 2^{-\Omega(t/\mu)}$.

Proof of Technical Lemma

- Each path has a unique history $t_1, \dots, t_\ell$ s. t. i-th node appears at time t_i.
- Prob(path with history $t_1, \dots, t_\ell$ created) $\le \left(\frac{1}{\mu}\right)^\ell$

Proof of Technical Lemma

- Each path has a unique history $t_1, \dots, t_\ell$ s. t. i-th node appears at time t_i.
- Prob(path with history $t_1, \dots, t_\ell$ created) $\le \left(\frac{1}{\mu}\right)^\ell$
- Consider at most t steps: at most $\binom{t}{\ell}$ choices for $0 \le t_1 < t_2 < \cdots < t_\ell \le t$.
- Prob($\exists$ path of length ℓ after $\ell\mu/3$ steps)

$$\le \binom{\ell\mu/3}{\ell}\left(\frac{1}{\mu}\right)^\ell \le \left(\frac{e\ell\mu}{3\ell}\right)^\ell \left(\frac{1}{\mu}\right)^\ell = 2^{-\Omega(\ell)}. \qquad \square$$

Application: General Lower Bound

Theorem (Witt, 2006)

Let f be a function with a unique optimum and $\mu = poly(n)$. Then the runtime of the $(\mu{+}1)$ EA on f is $\Omega(\mu n)$ with probability $1 - 2^{-\Omega(n)}$.

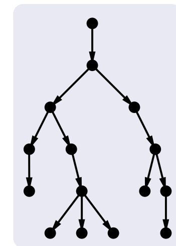
Application: General Lower Bound

Theorem (Witt, 2006)

Let f be a function with a unique optimum and $\mu = poly(n)$. Then the runtime of the $(\mu{+}1)$ EA on f is $\Omega(\mu n)$ with probability $1 - 2^{-\Omega(n)}$.

Proof idea:

- W. o. p.: after $\mu n/12$ steps: all paths in family trees have length $\leq n/4$.
- W. o. p.: initially, for all individuals: Hamming distance $\geq n/3$ from optimum.
- W. o. p.: $n/4$ mutations do not overcome Hamming distance $\geq n/3$.



RSHs for Combinatorial Optimisation

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

RSHs for Combinatorial Optimisation

- Analyse runtime and approximation quality on well-known combinatorial optimisation problems, e. g.,
  - sorting problems (is this an optimisation problem?),
  - shortest path problems,
  - Eulerian cycles,
  - mininum 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

- Analyse runtime and approximation quality on well-known combinatorial optimisation problems, e. g.,
  - sorting problems (is this an optimisation problem?),
  - shortest path problems,
  - Eulerian cycles,
  - mininum 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

This Tutorial

(1+1) EA for the Minimum Spanning Tree Problem

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

Fitness function: weight of tree/leading to trees for non-trees

(1+1) EA for the Minimum Spanning Tree Problem

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

Fitness function: weight of tree/leading to trees for non-trees

Observation: non-optimal trees improvable by exchanging just two edges $\rightarrow$ local change with expected factor $1 - 1/n$ for distance decrease from optimum

(1+1) EA for the Minimum Spanning Tree Problem

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

Fitness function: weight of tree/leading to trees for non-trees

Observation: non-optimal trees improvable by exchanging just two edges $\rightarrow$ local change with expected factor $1 - 1/n$ for distance decrease from optimum

Theorem (Neumann/Wegener, 2007)

The expected time until the (1+1) EA has created an MST is bounded by $O(n^4(\log n + \log w_{\max}))$.

(1+1) EA for the Minimum Spanning Tree Problem

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

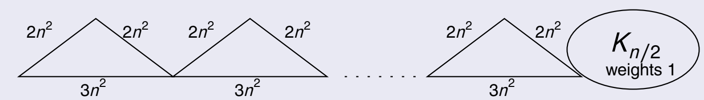
Fitness function: weight of tree/leading to trees for non-trees

Observation: non-optimal trees improvable by exchanging just two edges $\rightarrow$ local change with expected factor $1 - 1/n$ for distance decrease from optimum

Theorem (Neumann/Wegener, 2007)

The expected time until the (1+1) EA has created an MST is bounded by $O(n^4(\log n + \log w_{\max}))$.

A Tight Example



This Tutorial

1. The origins: example functions and toy problems
   - A simple toy problem: OneMax for (1+1) EA
   - An advanced example: population size for the (μ+1) EA
2. Combinatorial optimisation problems
   - (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

(1+1) EA for the Maximum Matching Problem

The Behaviour on Paths

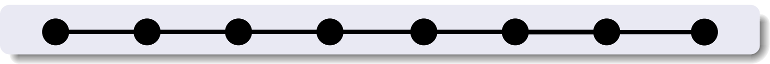
$n + 1$ nodes, n edges: bit string from $\{0,1\}^n$ selects edges

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

(1+1) EA for the Maximum Matching Problem
The Behaviour on Paths

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

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



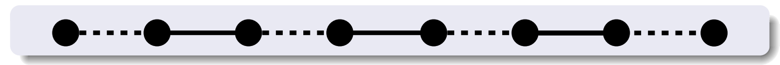
Theorem (Giel/Wegener, 2003)

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

(1+1) EA for the Maximum Matching Problem
The Behaviour on Paths (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 $\Theta(1/n^2)$.



(1+1) EA for the Maximum Matching Problem
The Behaviour on Paths (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

(1+1) EA for the Maximum Matching Problem
The Behaviour on Paths (2)

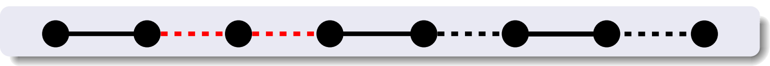
Proof idea:

- 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

(1+1) EA for the Maximum Matching Problem
The Behaviour on Paths (2)

Proof idea:

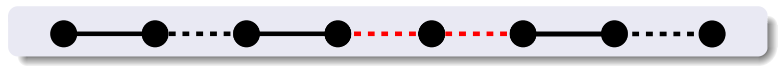
- Consider a second-best matching.
- Is there a free edge? Flip one bit! $\longrightarrow$ probability $\Theta(1/n)$.
- Else 2-bit flips $\longrightarrow$ probability $\Theta(1/n^2)$.
- Shorten augmenting path



(1+1) EA for the Maximum Matching Problem
The Behaviour on Paths (2)

Proof idea:

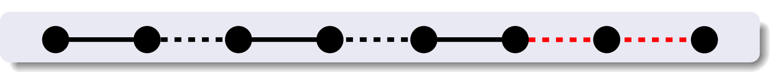
- Consider a second-best matching.
- Is there a free edge? Flip one bit! $\longrightarrow$ probability $\Theta(1/n)$.
- Else 2-bit flips $\longrightarrow$ probability $\Theta(1/n^2)$.
- Shorten augmenting path



(1+1) EA for the Maximum Matching Problem
The Behaviour on Paths (2)

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!



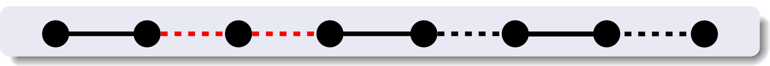
(1+1) EA for the Maximum Matching Problem
The Behaviour on Paths (2)

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!

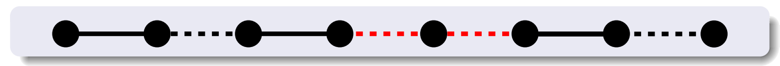
(1+1) EA for the Maximum Matching Problem
The Behaviour on Paths (2)

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



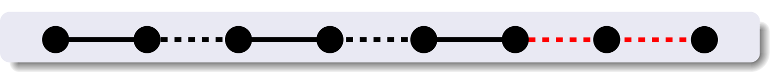
(1+1) EA for the Maximum Matching Problem
The Behaviour on Paths (2)

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



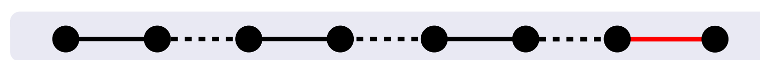
(1+1) EA for the Maximum Matching Problem
The Behaviour on Paths (2)

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



(1+1) EA for the Maximum Matching Problem
The Behaviour on Paths (2)

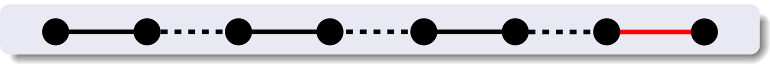
Proof idea:
- Consider a second-best matching.
- Is there a free edge? Flip one bit! → probability $\Theta(1/n)$.
- Else 2-bit flips → probability $\Theta(1/n^2)$.
- Shorten augmenting path
- Then flip the free edge!
- (1+1) EA follows the concept of an augmenting path!



(1+1) EA for the Maximum Matching Problem
The Behaviour on Paths (2)

Proof idea:

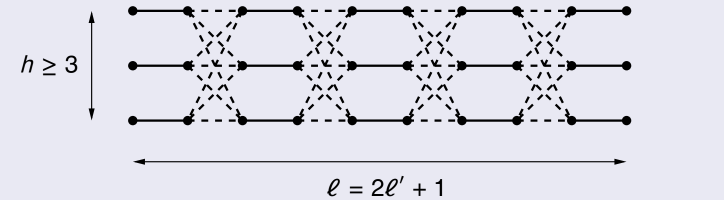
- Consider a second-best matching.
- Is there a free edge? Flip one bit! → probability $\Theta(1/n)$.
- Else 2-bit flips → probability $\Theta(1/n^2)$.
- Shorten augmenting path
- Then flip the free edge!
- (1+1) EA follows the concept of an augmenting path!



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

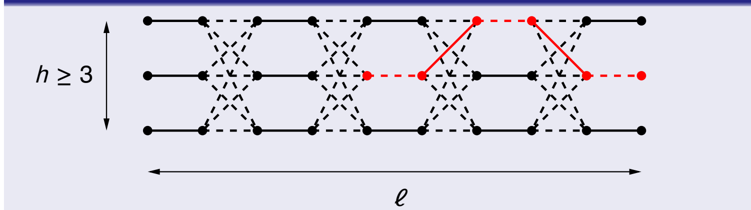
(1+1) EA for the Maximum Matching Problem
A Negative Result

Worst-case graph (Sasaki/Hajek, 1988)



(1+1) EA for the Maximum Matching Problem
A Negative Result

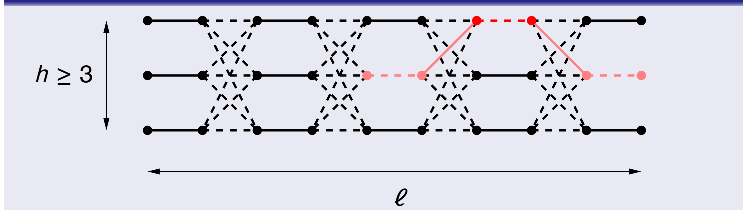
Worst-case graph (Sasaki/Hajek, 1988)



Augmenting path

(1+1) EA for the Maximum Matching Problem
A Negative Result

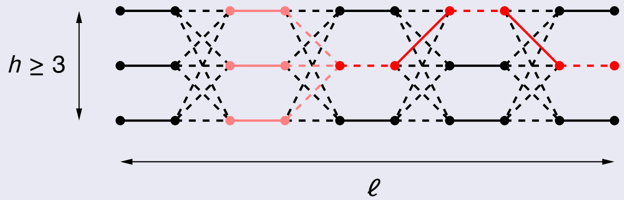
Worst-case graph (Sasaki/Hajek, 1988)



Augmenting path can get shorter

(1+1) EA for the Maximum Matching Problem
A Negative Result

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

(1+1) EA for the Maximum Matching Problem
(1+1) EA is a 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).

(1+1) EA for the Maximum Matching Problem
(1+1) EA is a 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) $\forall X \subseteq Y \in \mathcal{F} : X \in \mathcal{F}$, and
(iii) $\forall X, Y \in \mathcal{F}, |X| > |Y| : \exists x \in X \setminus Y$ with $Y \cup \{x\} \in \mathcal{F}$.

Adding a function $w : E \to \mathbb{N}$ 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) $\forall X \subseteq Y \in \mathcal{F} : X \in \mathcal{F}$, and
(iii) $\forall X, Y \in \mathcal{F}, |X| > |Y| : \exists x \in X \setminus Y$ with $Y \cup \{x\} \in \mathcal{F}$.

Adding a function $w : E \to \mathbb{N}$ 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) $\forall X \subseteq Y \in \mathcal{F} : X \in \mathcal{F}$, and
(iii) $\forall X, Y \in \mathcal{F}, |X| > |Y| : \exists x \in X \setminus Y$ with $Y \cup \{x\} \in \mathcal{F}$.

Adding a function $w : E \to \mathbb{N}$ 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

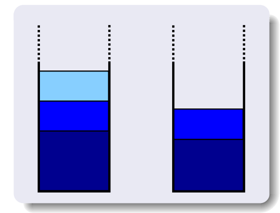
What about NP-hard problems? → Study approximation quality

(1+1) EA and the Partition Problem

What about NP-hard problems? → Study approximation quality

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

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

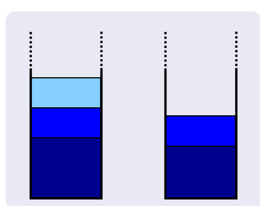


(1+1) EA and the Partition Problem

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,
- ...

(1+1) EA for the Partition Problem
Worst-Case Results

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 $O(n^2)$.

(1+1) EA for the Partition Problem
Worst-Case Results

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

(1+1) EA for the Partition Problem
Worst Case – PRAS by Parallelism

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.

(1+1) EA for the Partition Problem
Worst Case – PRAS by Parallelism

Theorem (Witt, 2005)

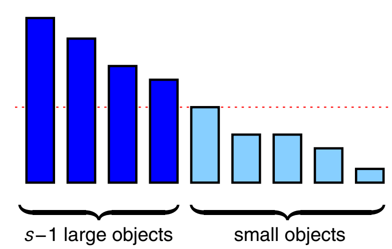
On any instance, the (1+1) EA with prob. $\geq 2^{-c\lceil 1/\varepsilon\rceil \ln(1/\varepsilon)}$ finds a $(1+\varepsilon)$-approximation within $O(n\ln(1/\varepsilon))$ steps.

- $2^{O(\lceil 1/\varepsilon\rceil \ln(1/\varepsilon))}$ 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!

(1+1) EA for the Partition Problem
Worst Case – PRAS by Parallelism (Proof Idea)

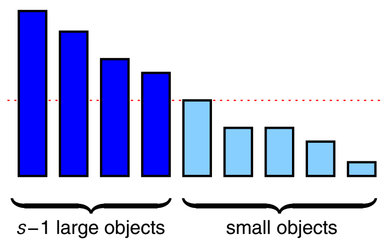
Set $s := \lceil \frac{2}{\varepsilon} \rceil$ and $w := \sum_{i=1}^{n} w_i$.

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



(1+1) EA for the Partition Problem
Worst Case – PRAS by Parallelism (Proof Idea)

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).

(1+1) EA for the Partition Problem
Average-Case Analyses

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.

(1+1) EA for the Partition Problem
Average-Case Analyses

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?

(1+1) EA for the Partition Problem
Partition Problem - Known Averge-Case Results

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).

(1+1) EA for the Partition Problem
Partition Problem - Known Averge-Case Results

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 $o(1)$?

(1+1) EA for the Partition Problem
New Result

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!

(1+1) EA for the Partition Problem
New Result

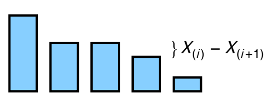
Theorem (Witt, 2005)

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:

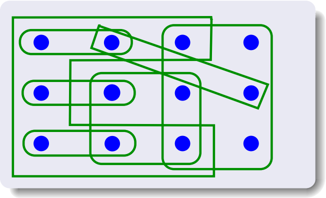
W. h. p.
$X_{(i)} - X_{(i+1)} = O((\log n)/n)$
for $i = \Omega(n)$.



This Tutorial

The Set Cover Problem

Another NP-hard problem

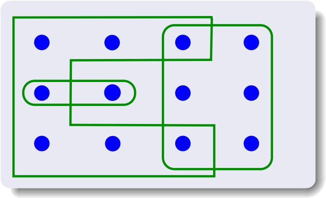


Given:

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

The Set Cover Problem

Another NP-hard problem



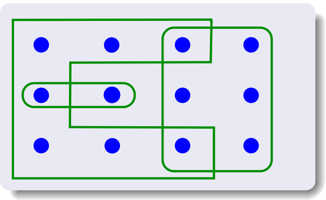
Given:

- 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 $\bigcup_{j=1}^{k} C_{i_j} = S$.

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

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

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 $f(x') \le f(z)$ 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 k: SEMO covers k elements at cost $\le \sum_{i=k+1}^{|S|} \frac{\mathrm{OPT}}{i}$.

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 k: SEMO covers k elements at cost $\le \sum_{i=k+1}^{|S|} \frac{\mathrm{OPT}}{i}$.
- 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 $\sum_{i=1}^{|S|} 1/i \le \ln|S| + 1$.

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 k: SEMO covers k elements at cost $\le \sum_{i=k+1}^{|S|} \frac{\mathrm{OPT}}{i}$.
- 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 $\sum_{i=1}^{|S|} 1/i \le \ln|S| + 1$.

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
   - An advanced example: population size for the (μ+1) EA
2. Combinatorial optimisation problems
   - (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.

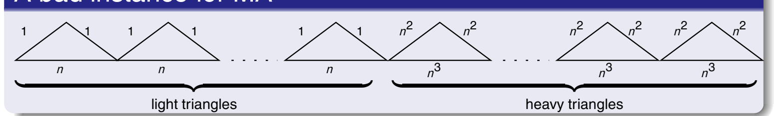
Simulated Annealing Beats Metropolis in Combinatorial Optimisation

Jerrum/Sinclair (1996)

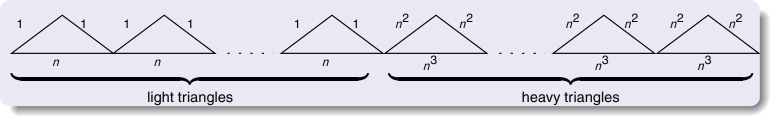
"It remains an outstanding open problem to exhibit a natural example in which simulated annealing with any non-trivial cooling schedule provably outperforms the Metropolis algorithm at a carefully chosen fixed value" of the temperature.

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

A bad instance for MA



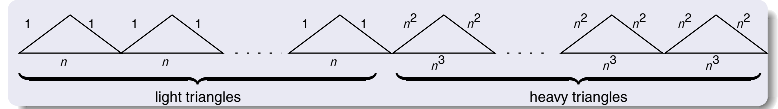
Simulated Annealing Beats Metropolis in Combinatorial Optimisation
Results



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 $1 - O(1/\mathrm{poly}(n))$.

Simulated Annealing Beats Metropolis in Combinatorial Optimisation
Results



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 $1 - O(1/\mathrm{poly}(n))$.

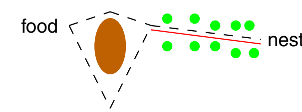
Proof idea: need different temperatures to optimise all triangles.

This Tutorial

1. The origins: example functions and toy problems
   - A simple toy problem: OneMax for (1+1) EA
   - An advanced example: population size for the $(\mu+1)$ EA
2. Combinatorial optimisation problems
   - (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 Colony Optimisation — A Modern Search Heuristic
Background and Motivation

Ant colonies in nature

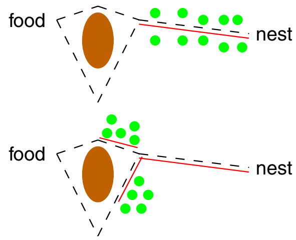


- find shortest paths in an unknown environment
- using communication via pheromone trails
- show adaptive behaviour

Ant Colony Optimisation — A Modern Search Heuristic
Background and Motivation

Ant colonies in nature

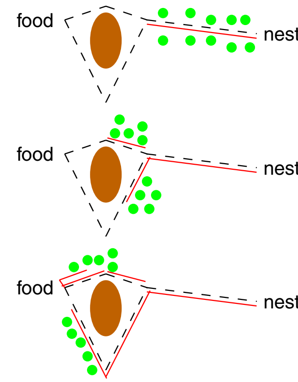
- find shortest paths in an unknown environment
- using communication via pheromone trails
- show adaptive behaviour



Ant Colony Optimisation — A Modern Search Heuristic
Background and Motivation

Ant colonies in nature

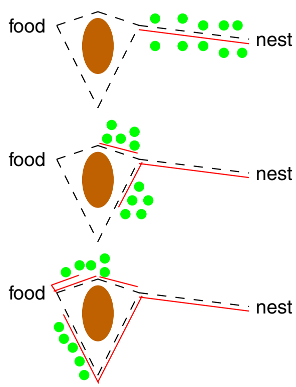
- find shortest paths in an unknown environment
- using communication via pheromone trails
- show adaptive behaviour



Ant Colony Optimisation — A Modern Search Heuristic
Background and Motivation

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

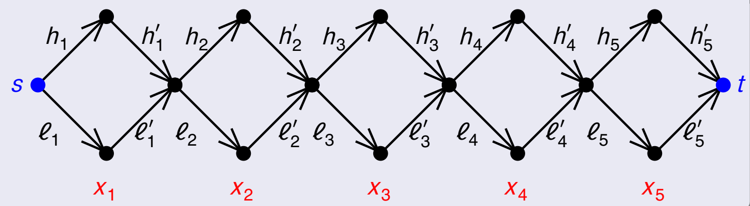
1-ANT for Pseudo-Boolean Optimisation

1-ANT

- Simple ACO algorithm
- Previously studied w. r. t. convergence
- Find maximum for pseudo-Boolean function $f\colon \{0,1\}^n \to \mathbb{R}$

1-ANT for Pseudo-Boolean Optimisation

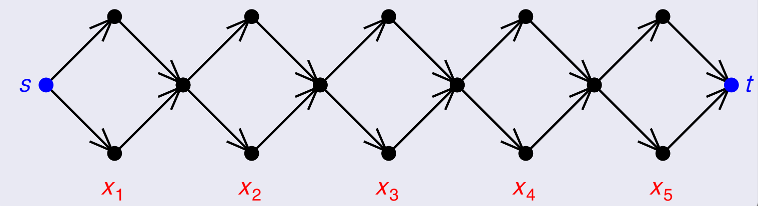
Construction graph ($n = 5$)



- Pheromone values $\tau(e)$ for all $4n$ edges e
- Ant constructs random path $P(x)$ from s to t.
- Edge h_i is taken with probability $\tau(h_i)/(\tau(h_i) + \tau(\ell_i))$, accordingly for ℓ_i.
- Walk constructs bitstring $x \in \{0, 1\}^n$.

1-ANT for Pseudo-Boolean Optimisation

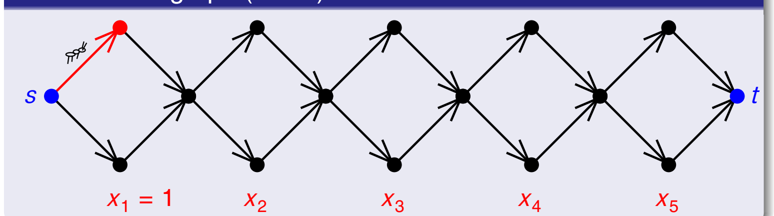
Construction graph ($n = 5$)



- Pheromone values $\tau(e)$ for all $4n$ edges e
- Ant constructs random path $P(x)$ from s to t.
- Edge h_i is taken with probability $\tau(h_i)/(\tau(h_i) + \tau(\ell_i))$, accordingly for ℓ_i.
- Walk constructs bitstring $x \in \{0, 1\}^n$.

1-ANT for Pseudo-Boolean Optimisation

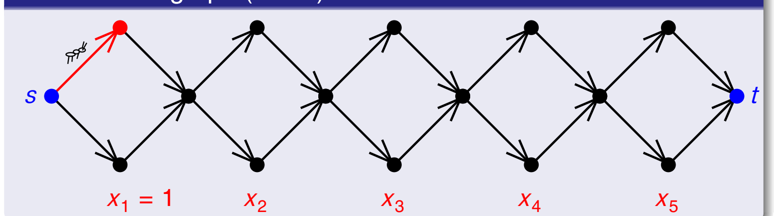
Construction graph ($n = 5$)



- Pheromone values $\tau(e)$ for all $4n$ edges e
- Ant constructs random path $P(x)$ from s to t.
- Edge h_i is taken with probability $\tau(h_i)/(\tau(h_i) + \tau(\ell_i))$, accordingly for ℓ_i.
- Walk constructs bitstring $x \in \{0, 1\}^n$.

1-ANT for Pseudo-Boolean Optimisation

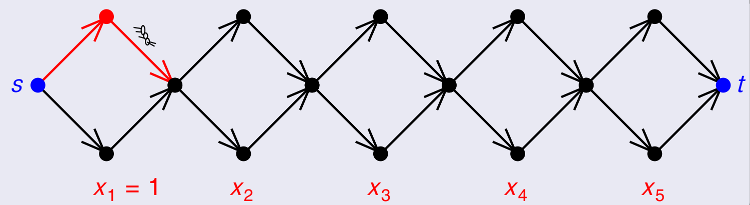
Construction graph ($n = 5$)



- Pheromone values $\tau(e)$ for all $4n$ edges e
- Ant constructs random path $P(x)$ from s to t.
- Edge h_i is taken with probability $\tau(h_i)/(\tau(h_i) + \tau(\ell_i))$, accordingly for ℓ_i.
- Walk constructs bitstring $x \in \{0, 1\}^n$.

1-ANT for Pseudo-Boolean Optimisation

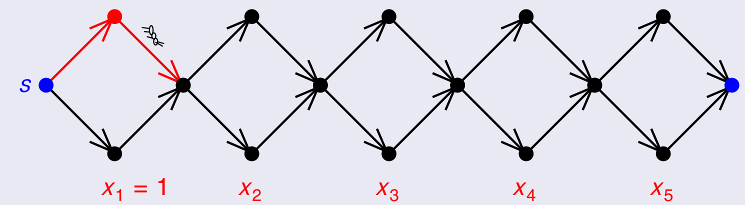
Construction graph ($n = 5$)



- Pheromone values $\tau(e)$ for all $4n$ edges e
- Ant constructs random path $P(x)$ from s to t.
- Edge h_i is taken with probability $\tau(h_i)/(\tau(h_i) + \tau(\ell_i))$, accordingly for ℓ_i.
- Walk constructs bitstring $x \in \{0, 1\}^n$.

1-ANT for Pseudo-Boolean Optimisation

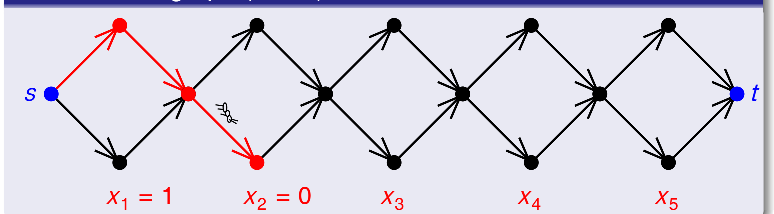
Construction graph ($n = 5$)



- Pheromone values $\tau(e)$ for all $4n$ edges e
- Ant constructs random path $P(x)$ from s to t.
- Edge h_i is taken with probability $\tau(h_i)/(\tau(h_i) + \tau(\ell_i))$, accordingly for ℓ_i.
- Walk constructs bitstring $x \in \{0, 1\}^n$.

1-ANT for Pseudo-Boolean Optimisation

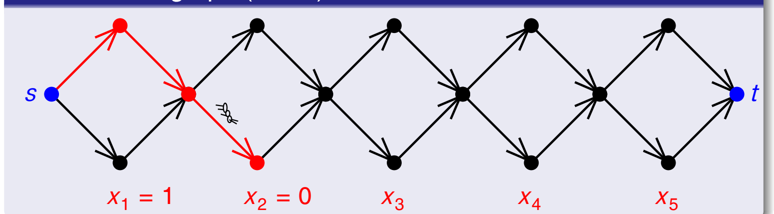
Construction graph ($n = 5$)



- Pheromone values $\tau(e)$ for all $4n$ edges e
- Ant constructs random path $P(x)$ from s to t.
- Edge h_i is taken with probability $\tau(h_i)/(\tau(h_i) + \tau(\ell_i))$, accordingly for ℓ_i.
- Walk constructs bitstring $x \in \{0, 1\}^n$.

1-ANT for Pseudo-Boolean Optimisation

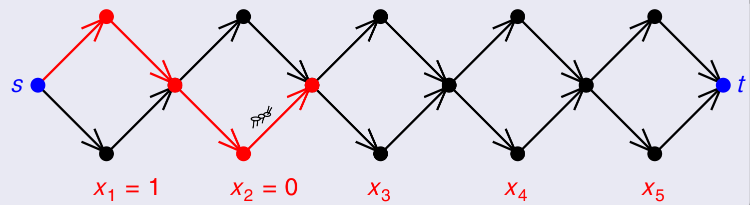
Construction graph ($n = 5$)



- Pheromone values $\tau(e)$ for all $4n$ edges e
- Ant constructs random path $P(x)$ from s to t.
- Edge h_i is taken with probability $\tau(h_i)/(\tau(h_i) + \tau(\ell_i))$, accordingly for ℓ_i.
- Walk constructs bitstring $x \in \{0, 1\}^n$.

1-ANT for Pseudo-Boolean Optimisation

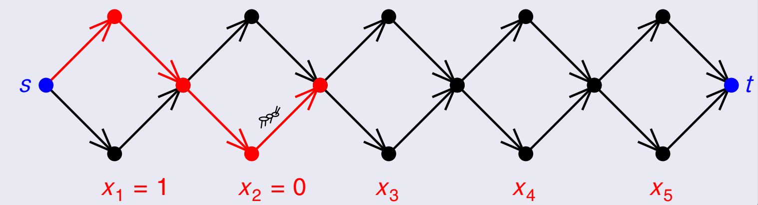
Construction graph ($n = 5$)



- Pheromone values $\tau(e)$ for all $4n$ edges e
- Ant constructs random path $P(x)$ from s to t.
- Edge h_i is taken with probability $\tau(h_i)/(\tau(h_i) + \tau(\ell_i))$, accordingly for ℓ_i.
- Walk constructs bitstring $x \in \{0, 1\}^n$.

1-ANT for Pseudo-Boolean Optimisation

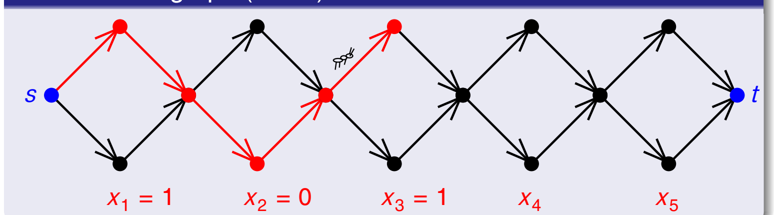
Construction graph ($n = 5$)



- Pheromone values $\tau(e)$ for all $4n$ edges e
- Ant constructs random path $P(x)$ from s to t.
- Edge h_i is taken with probability $\tau(h_i)/(\tau(h_i) + \tau(\ell_i))$, accordingly for ℓ_i.
- Walk constructs bitstring $x \in \{0, 1\}^n$.

1-ANT for Pseudo-Boolean Optimisation

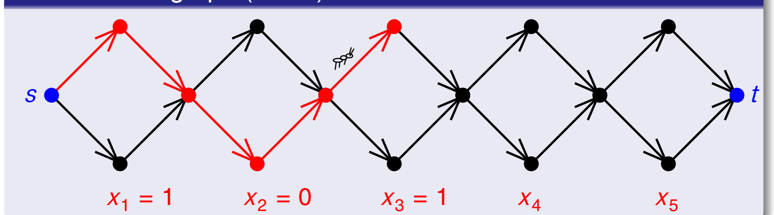
Construction graph ($n = 5$)



- Pheromone values $\tau(e)$ for all $4n$ edges e
- Ant constructs random path $P(x)$ from s to t.
- Edge h_i is taken with probability $\tau(h_i)/(\tau(h_i) + \tau(\ell_i))$, accordingly for ℓ_i.
- Walk constructs bitstring $x \in \{0, 1\}^n$.

1-ANT for Pseudo-Boolean Optimisation

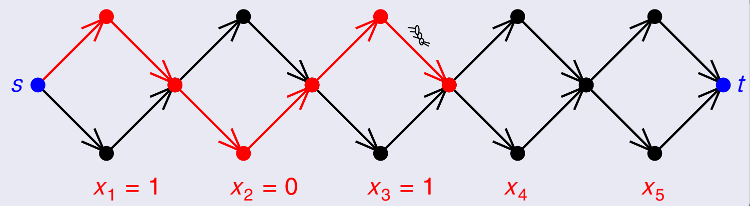
Construction graph ($n = 5$)



- Pheromone values $\tau(e)$ for all $4n$ edges e
- Ant constructs random path $P(x)$ from s to t.
- Edge h_i is taken with probability $\tau(h_i)/(\tau(h_i) + \tau(\ell_i))$, accordingly for ℓ_i.
- Walk constructs bitstring $x \in \{0, 1\}^n$.

1-ANT for Pseudo-Boolean Optimisation

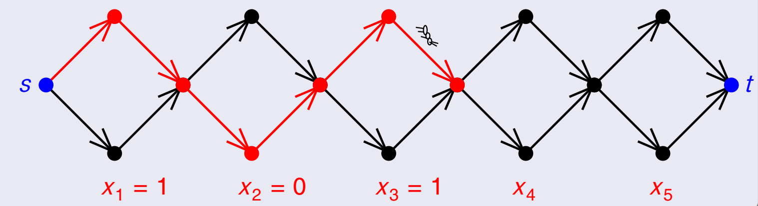
Construction graph ($n = 5$)



- Pheromone values $\tau(e)$ for all $4n$ edges e
- Ant constructs random path $P(x)$ from s to t.
- Edge h_i is taken with probability $\tau(h_i)/(\tau(h_i) + \tau(\ell_i))$, accordingly for ℓ_i.
- Walk constructs bitstring $x \in \{0, 1\}^n$.

1-ANT for Pseudo-Boolean Optimisation

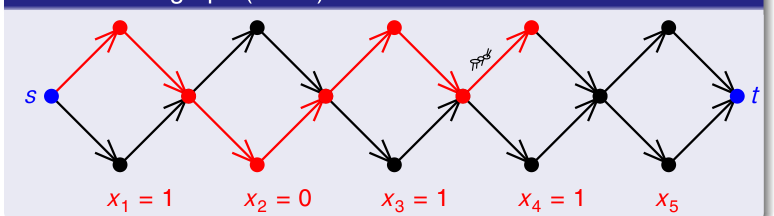
Construction graph ($n = 5$)



- Pheromone values $\tau(e)$ for all $4n$ edges e
- Ant constructs random path $P(x)$ from s to t.
- Edge h_i is taken with probability $\tau(h_i)/(\tau(h_i) + \tau(\ell_i))$, accordingly for ℓ_i.
- Walk constructs bitstring $x \in \{0, 1\}^n$.

1-ANT for Pseudo-Boolean Optimisation

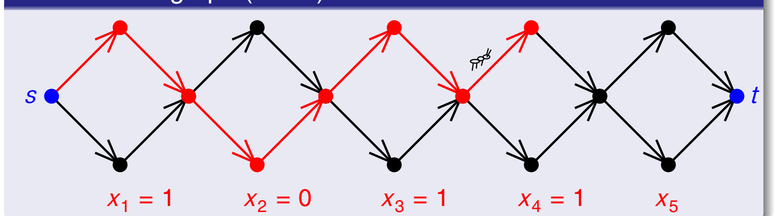
Construction graph ($n = 5$)



- Pheromone values $\tau(e)$ for all $4n$ edges e
- Ant constructs random path $P(x)$ from s to t.
- Edge h_i is taken with probability $\tau(h_i)/(\tau(h_i) + \tau(\ell_i))$, accordingly for ℓ_i.
- Walk constructs bitstring $x \in \{0, 1\}^n$.

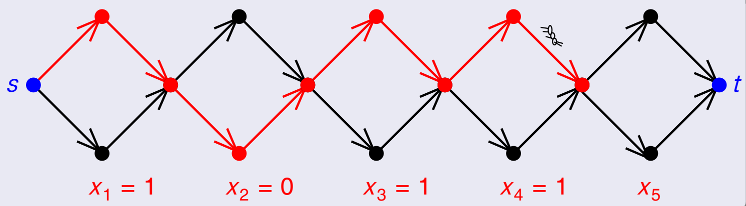
1-ANT for Pseudo-Boolean Optimisation

Construction graph ($n = 5$)



- Pheromone values $\tau(e)$ for all $4n$ edges e
- Ant constructs random path $P(x)$ from s to t.
- Edge h_i is taken with probability $\tau(h_i)/(\tau(h_i) + \tau(\ell_i))$, accordingly for ℓ_i.
- Walk constructs bitstring $x \in \{0, 1\}^n$.

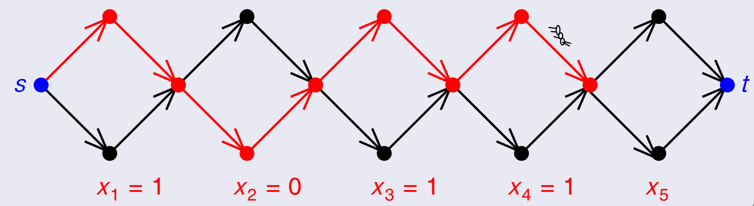
Construction graph ($n = 5$)



- Pheromone values $\tau(e)$ for all $4n$ edges e
- Ant constructs random path $P(x)$ from s to t.
- Edge h_i is taken with probability $\tau(h_i)/(\tau(h_i) + \tau(\ell_i))$, accordingly for ℓ_i.
- Walk constructs bitstring $x \in \{0, 1\}^n$.

1-ANT for Pseudo-Boolean Optimisation

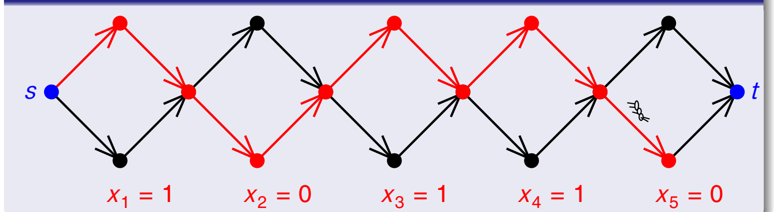
Construction graph ($n = 5$)



- Pheromone values $\tau(e)$ for all $4n$ edges e
- Ant constructs random path $P(x)$ from s to t.
- Edge h_i is taken with probability $\tau(h_i)/(\tau(h_i) + \tau(\ell_i))$, accordingly for ℓ_i.
- Walk constructs bitstring $x \in \{0, 1\}^n$.

1-ANT for Pseudo-Boolean Optimisation

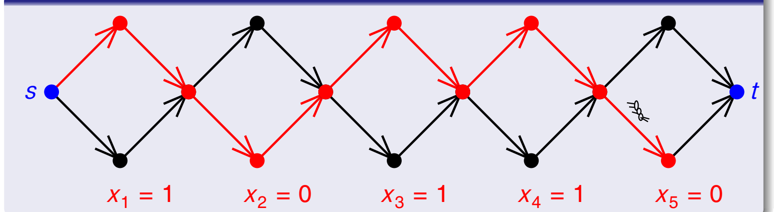
Construction graph ($n = 5$)



- Pheromone values $\tau(e)$ for all $4n$ edges e
- Ant constructs random path $P(x)$ from s to t.
- Edge h_i is taken with probability $\tau(h_i)/(\tau(h_i) + \tau(\ell_i))$, accordingly for ℓ_i.
- Walk constructs bitstring $x \in \{0, 1\}^n$.

1-ANT for Pseudo-Boolean Optimisation

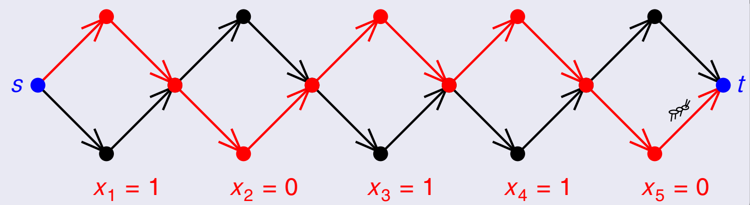
Construction graph ($n = 5$)



- Pheromone values $\tau(e)$ for all $4n$ edges e
- Ant constructs random path $P(x)$ from s to t.
- Edge h_i is taken with probability $\tau(h_i)/(\tau(h_i) + \tau(\ell_i))$, accordingly for ℓ_i.
- Walk constructs bitstring $x \in \{0, 1\}^n$.

1-ANT for Pseudo-Boolean Optimisation

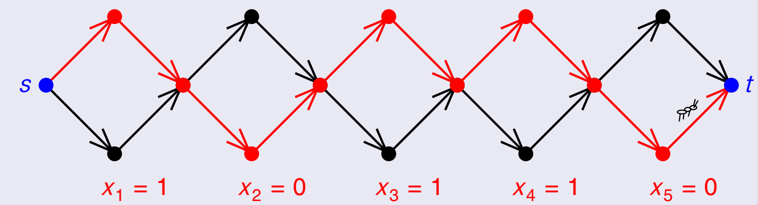
Construction graph ($n = 5$)



- Pheromone values $\tau(e)$ for all $4n$ edges e
- Ant constructs random path $P(x)$ from s to t.
- Edge h_i is taken with probability $\tau(h_i)/(\tau(h_i) + \tau(\ell_i))$, accordingly for ℓ_i.
- Walk constructs bitstring $x \in \{0, 1\}^n$.

1-ANT for Pseudo-Boolean Optimisation

Construction graph ($n = 5$)



- Pheromone values $\tau(e)$ for all $4n$ edges e
- Ant constructs random path $P(x)$ from s to t.
- Edge h_i is taken with probability $\tau(h_i)/(\tau(h_i) + \tau(\ell_i))$, accordingly for ℓ_i.
- Walk constructs bitstring $x \in \{0, 1\}^n$.

1-ANT – Outline

Conventions

- Pheromone values = probabilities
- Upper and lower bounds for pheromone values
- Runtime = # constructed solutions until optimum found

1-ANT – Outline

Conventions

- Pheromone values = probabilities
- Upper and lower bounds for pheromone values
- Runtime = # constructed solutions until optimum found

Algorithm 1-ANT for functions $f\colon \{0, 1\}^n \to \mathbb{R}$

- Set $\tau(e) = \frac{1}{2}$ for all edges e.
- Construct x (and $P(x)$), update pheromone; set $x^* := x$.
- Repeat
  - Construct x (and $P(x)$).
  - If $f(x) \geq f(x^*)$, update pheromone and set $x^* := x$.

1-ANT – Pheromone Update

- Crucial parameter: evaporation factor ρ, $0 \le \rho \le 1$
- Edge e is updated according to

$$e \in P(x) \quad \Rightarrow \quad \tau(e) := \min\left\{(1-\rho)\cdot\tau(e)+\rho,\, 1-\frac{1}{n}\right\}.$$
$$e \notin P(x) \quad \Rightarrow \quad \tau(e) := \max\left\{(1-\rho)\cdot\tau(e),\, \frac{1}{n}\right\}.$$

1-ANT – Pheromone Update

- Crucial parameter: evaporation factor ρ, $0 \le \rho \le 1$
- Edge e is updated according to

$$e \in P(x) \quad \Rightarrow \quad \tau(e) := \min\left\{(1-\rho)\cdot\tau(e)+\rho,\, 1-\frac{1}{n}\right\}.$$
$$e \notin P(x) \quad \Rightarrow \quad \tau(e) := \max\left\{(1-\rho)\cdot\tau(e),\, \frac{1}{n}\right\}.$$

- $\tau(h_i) + \tau(\ell_i) = 1$ for $1 \le i \le n$, i. e., probabilities
- Upper and lower bounds ensure that all probabilities in $[1/n, 1-1/n]$.

1-ANT: Runtime Analyses

- Simple but crucial: 1-ANT generalises (1+1) EA (just choose ρ large enough to keep all pheromone values in $\{1/n, 1-1/n\}$).

1-ANT: Runtime Analyses

- Simple but crucial: 1-ANT generalises (1+1) EA (just choose ρ large enough to keep all pheromone values in $\{1/n, 1-1/n\}$).
- Old friends return: example functions
- Results depending on ρ:

	superpol. runtime	poly. runtime
OneMax	$\rho = o(1/\log n)$	$\rho = 1 - O(n^{-\varepsilon})$
LeadingOnes	$\rho \le c_1/\log n$	$\rho \ge c_2/\log n$
BinVal	$\rho \le c_1/\log n$	$\rho \ge c_2/\log n$

(Neumann/Witt, 2006; Doerr/Neumann/Sudholt/Witt, 2007; Doerr/Johannsen, 2007)

- Phase transitions: 1-ANT is not robust w. r. t. ρ

1-ANT: Runtime Analyses

- Simple but crucial: 1-ANT generalises (1+1) EA (just choose ρ large enough to keep all pheromone values in $\{1/n, 1-1/n\}$).
- Old friends return: example functions
- Results depending on ρ:

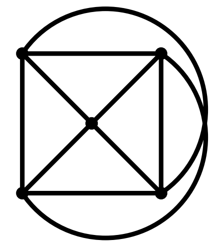
	superpol. runtime	poly. runtime
OneMax	$\rho = o(1/\log n)$	$\rho = 1 - O(n^{-\varepsilon})$
LeadingOnes	$\rho \le c_1/\log n$	$\rho \ge c_2/\log n$
BinVal	$\rho \le c_1/\log n$	$\rho \ge c_2/\log n$

(Neumann/Witt, 2006; Doerr/Neumann/Sudholt/Witt, 2007; Doerr/Johannsen, 2007)

- Phase transitions: 1-ANT is not robust w. r. t. ρ
- Interesting for proofs: need inverse of concentration inequalities (old result by Hoeffding)

1-ANT for the MST problem
1st Construction Graph

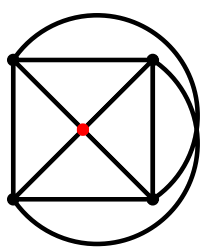
- Algorithm by Broder (1989): uniformly generate spanning trees by random walks on graphs
- Random walk uniformly chooses a neighbour. If unvisited, add edge to spanning tree
- Algorithm stops after expected $O(n^3)$ steps (cover time).



Selected edges obtain higher, (but not too high) pheromone values
→ next constructed tree similar, but also likely to be better

1-ANT for the MST problem
1st Construction Graph

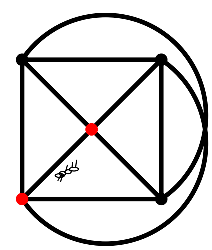
- Algorithm by Broder (1989): uniformly generate spanning trees by random walks on graphs
- Random walk uniformly chooses a neighbour. If unvisited, add edge to spanning tree
- Algorithm stops after expected $O(n^3)$ steps (cover time).



Selected edges obtain higher, (but not too high) pheromone values
→ next constructed tree similar, but also likely to be better

1-ANT for the MST problem
1st Construction Graph

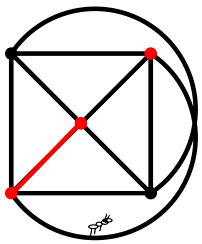
- Algorithm by Broder (1989): uniformly generate spanning trees by random walks on graphs
- Random walk uniformly chooses a neighbour. If unvisited, add edge to spanning tree
- Algorithm stops after expected $O(n^3)$ steps (cover time).



Selected edges obtain higher, (but not too high) pheromone values
→ next constructed tree similar, but also likely to be better

1-ANT for the MST problem
1st Construction Graph

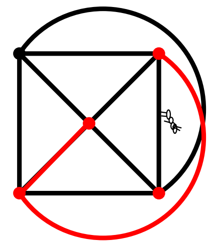
- Algorithm by Broder (1989): uniformly generate spanning trees by random walks on graphs
- Random walk uniformly chooses a neighbour. If unvisited, add edge to spanning tree
- Algorithm stops after expected $O(n^3)$ steps (cover time).



Selected edges obtain higher, (but not too high) pheromone values $\rightarrow$ next constructed tree similar, but also likely to be better

1-ANT for the MST problem
1st Construction Graph

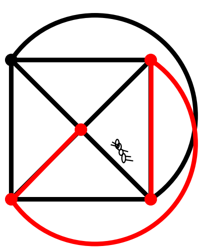
- Algorithm by Broder (1989): uniformly generate spanning trees by random walks on graphs
- Random walk uniformly chooses a neighbour. If unvisited, add edge to spanning tree
- Algorithm stops after expected $O(n^3)$ steps (cover time).



Selected edges obtain higher, (but not too high) pheromone values $\rightarrow$ next constructed tree similar, but also likely to be better

1-ANT for the MST problem
1st Construction Graph

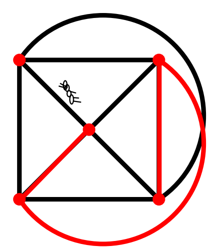
- Algorithm by Broder (1989): uniformly generate spanning trees by random walks on graphs
- Random walk uniformly chooses a neighbour. If unvisited, add edge to spanning tree
- Algorithm stops after expected $O(n^3)$ steps (cover time).



Selected edges obtain higher, (but not too high) pheromone values $\rightarrow$ next constructed tree similar, but also likely to be better

1-ANT for the MST problem
1st Construction Graph

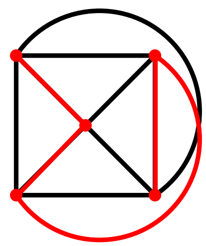
- Algorithm by Broder (1989): uniformly generate spanning trees by random walks on graphs
- Random walk uniformly chooses a neighbour. If unvisited, add edge to spanning tree
- Algorithm stops after expected $O(n^3)$ steps (cover time).



Selected edges obtain higher, (but not too high) pheromone values $\rightarrow$ next constructed tree similar, but also likely to be better

1-ANT for the MST problem
1st Construction Graph

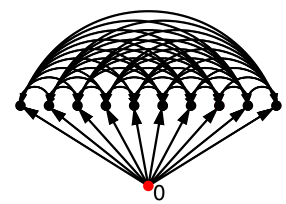
- Algorithm by Broder (1989): uniformly generate spanning trees by random walks on graphs
- Random walk uniformly chooses a neighbour. If unvisited, add edge to spanning tree
- Algorithm stops after expected $O(n^3)$ steps (cover time).



Selected edges obtain higher, (but not too high) pheromone values $\longrightarrow$ next constructed tree similar, but also likely to be better

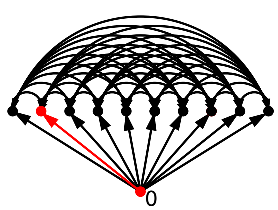
1-ANT for the MST problem
2nd Construction Graph

- Canonical construction graphs for a combinatorial optimisation problem identifies components with nodes and possible combinations with selectable edges.
- Here: components = edges $\longrightarrow$ canonical construction graph $C(G) = (N, A)$ with $N = \{0, \dots, m\}$ (start node 0) and $A = \{(i, j) \mid 0 \le i \le m, 1 \le j \le m, i \ne j\}$.



1-ANT for the MST problem
2nd Construction Graph

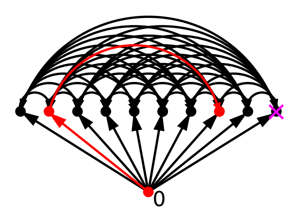
- Canonical construction graphs for a combinatorial optimisation problem identifies components with nodes and possible combinations with selectable edges.
- Here: components = edges $\longrightarrow$ canonical construction graph $C(G) = (N, A)$ with $N = \{0, \dots, m\}$ (start node 0) and $A = \{(i, j) \mid 0 \le i \le m, 1 \le j \le m, i \ne j\}$.



For path $v_1, \dots, v_k$ allowed neighbourhood $N(v_1, \dots, v_k) :=$ $(E \setminus \{v_1, \dots, v_k\}) \setminus \{e \in E \mid$ $(V, \{v_1, \dots, v_k, e\})$ contains cycle$\}$ (problem-specific aspect of ACO).

1-ANT for the MST problem
2nd Construction Graph

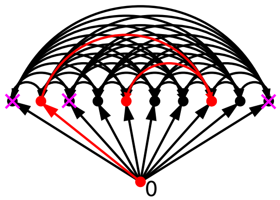
- Canonical construction graphs for a combinatorial optimisation problem identifies components with nodes and possible combinations with selectable edges.
- Here: components = edges $\longrightarrow$ canonical construction graph $C(G) = (N, A)$ with $N = \{0, \dots, m\}$ (start node 0) and $A = \{(i, j) \mid 0 \le i \le m, 1 \le j \le m, i \ne j\}$.



For path $v_1, \dots, v_k$ allowed neighbourhood $N(v_1, \dots, v_k) :=$ $(E \setminus \{v_1, \dots, v_k\}) \setminus \{e \in E \mid$ $(V, \{v_1, \dots, v_k, e\})$ contains cycle$\}$ (problem-specific aspect of ACO).

1-ANT for the MST problem
2nd Construction Graph

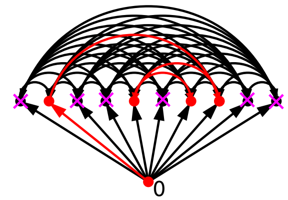
- Canonical construction graphs for a combinatorial optimisation problem identifies components with nodes and possible combinations with selectable edges.
- Here: components = edges → canonical construction graph $C(G) = (N, A)$ with $N = \{0, \dots, m\}$ (start node 0) and $A = \{(i, j) \mid 0 \le i \le m, 1 \le j \le m, i \ne j\}$.



For path $v_1, \dots, v_k$ allowed neighbourhood $N(v_1, \dots, v_k) := (E \setminus \{v_1, \dots, v_k\}) \setminus \{e \in E \mid (V, \{v_1, \dots, v_k, e\})$ contains cycle$\}$ (problem-specific aspect of ACO).

1-ANT for the MST problem
2nd Construction Graph

- Canonical construction graphs for a combinatorial optimisation problem identifies components with nodes and possible combinations with selectable edges.
- Here: components = edges → canonical construction graph $C(G) = (N, A)$ with $N = \{0, \dots, m\}$ (start node 0) and $A = \{(i, j) \mid 0 \le i \le m, 1 \le j \le m, i \ne j\}$.



For path $v_1, \dots, v_k$ allowed neighbourhood $N(v_1, \dots, v_k) := (E \setminus \{v_1, \dots, v_k\}) \setminus \{e \in E \mid (V, \{v_1, \dots, v_k, e\})$ contains cycle$\}$ (problem-specific aspect of ACO).

1-ANT for the MST Problem
Results

Theorem (Neumann/Witt, 2006)

The expected number of constructed solutions until the 1-ANT with the 1st construction graph finds an MST is $O(n^6(\log n + \log w_{\max}))$

The expected runtime of the construction procedure is $O(n^3)$.

1-ANT for the MST Problem
Results

Theorem (Neumann/Witt, 2006)

The expected number of constructed solutions until the 1-ANT with the 1st construction graph finds an MST is $O(n^6(\log n + \log w_{\max}))$

The expected runtime of the construction procedure is $O(n^3)$.

Theorem (Neumann/Witt, 2006)

The expected number of constructed solutions until the 1-ANT with the 2nd construction graph finds an MST is $O(mn(\log n + \log w_{\max}))$.

Better than the (1+1) EA!

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

→ The analysis of RSHs is an exciting research direction.

Thank you!

Selected Literature I

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

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

Selected Literature II

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

Selected Literature III

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 '01, 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

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

Selected Literature IV

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 '93

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

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

J. Reichel and M. Skutella (2007):
Evolutionary algorithms and matroid optimization problems.
Proc. of GECCO '07, 947–954, ACM Press

Selected Literature V

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

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 (μ+1) EA on simple pseudo-boolean functions.
Evolutionary Computation, 14(1), 65–86