Bioinspired Computation in Combinatorial Optimization – Algorithms and Their Computational Complexity

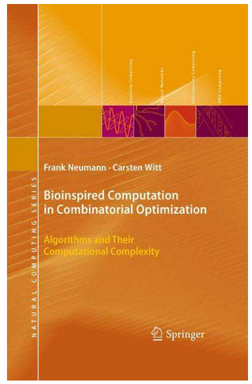
Carsten Witt

Technical University of Denmark
www.compute.dtu.dk/~cawi

Tutorial at GECCO 2014

GECCO'14 Companion, July 12–16, 2014, Vancouver, BC, Canada.
ACM 978-1-4503-2662-9/14/07.

Parts of the material used with kind permission by Frank Neumann.



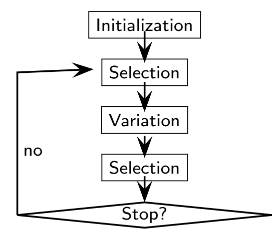
Book available at www.bioinspiredcomputation.com

GECCO '14, Jul 12–16 2014, Vancouver, BC, Canada
ACM 978-1-4503-2881-4/14/07.
http://dx.doi.org/10.1145/2598394.2605353

Evolutionary Algorithms and Other Search Heuristics

Most famous search heuristic: Evolutionary Algorithms (EAs)

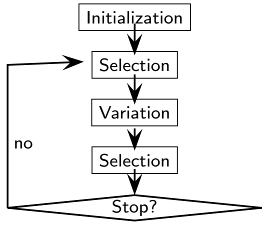
- a bio-inspired heuristic
- paradigm: evolution in nature, "survival of the fittest"



Evolutionary Algorithms and Other Search Heuristics

Most famous search heuristic: Evolutionary Algorithms (EAs)

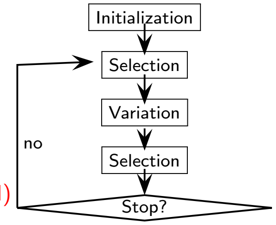
- a bio-inspired heuristic
- paradigm: evolution in nature, "survival of the fittest"
- actually it's only an algorithm, a randomized search heuristic (RSH)



Evolutionary Algorithms and Other Search Heuristics

Most famous search heuristic: Evolutionary Algorithms (EAs)

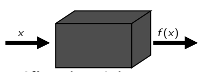
- a bio-inspired heuristic
- paradigm: evolution in nature, "survival of the fittest"
- actually it's only an algorithm, a randomized search heuristic (RSH)



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

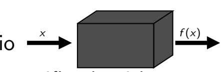
Optimize $f\colon \{0,1\}^n \to \mathbb{R}$

Why Do We Consider Randomized 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 Randomized Search Heuristics
- They are surprisingly successful.

Why Do We Consider Randomized 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 Randomized Search Heuristics
- They are surprisingly successful.

Point of view

Want 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)
- Randomized Local Search (RLS)
- Metropolis Algorithm/Simulated Annealing (MA/SA)
- Ant Colony Optimization (ACO)
- Particle Swarm Optimization (PSO)
- ...

First of all: define the simple ones

The Most Basic RSHs

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

(1+1) EA

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

RLS

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

MA

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

SA

1. Choose $x_0 \in \{0,1\}^n$ uniformly at random.
2. For $t := 0, \dots, \infty$
   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 studied here: convergence, local progress, models of EAs (e. g., infinite populations), ...
- Treat RSHs as randomized algorithm!
- Analyze their “runtime” (computational complexity) on selected problems

What Kind of Theory Are We Interested in?

- Not studied here: convergence, local progress, models of EAs (e. g., infinite populations), ...
- Treat RSHs as randomized algorithm!
- Analyze their “runtime” (computational complexity) on selected problems

Definition

Let RSH A optimize 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 amortized 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 amortized analysis
- ...

Adapt tools from the analysis of randomized 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
- ...

High-quality results, but limited to SA/MA (nothing about EAs) and hard to generalize.

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
- ...

High-quality results, but limited to SA/MA (nothing about EAs) and hard to generalize.

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 optimization problems
   - Minimum spanning trees
   - Maximum matchings
   - Shortest paths
   - Makespan scheduling
   - SA beats MA in combinatorial optimization
3. End

How the Systematic Research Began — Toy Problems

Simple example functions (test functions)

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

Goal: derive first runtime bounds and methods

How the Systematic Research Began — Toy Problems

Simple example functions (test functions)

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

Agenda

1. The origins: example functions and toy problems
   - A simple toy problem: OneMax for (1+1) EA
2. Combinatorial optimization problems
   - Minimum spanning trees
   - Maximum matchings
   - Shortest paths
   - Makespan scheduling
   - SA beats MA in combinatorial optimization
3. End

Example: OneMax

Theorem

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

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

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 all linear functions!).
- Bound the optimization time for linear functions ($O(n \log n)$).
- Monotone functions can be difficult.
- Optimal population size (often 1!)
- Crossover vs. no crossover $\rightarrow$ Real Royal Road Functions
- Multistarts vs. populations
- Frequent restarts vs. long runs
- Dynamic schedules
- ...

RSHs for Combinatorial Optimization

- Analysis of runtime and approximation quality on well-known combinatorial optimization problems, e. g.,
  - sorting problems (is this an optimization problem?),
  - covering problems,
  - cutting problems,
  - subsequence problems,
  - traveling salesperson problem,
  - Eulerian cycles,
  - shortest path problems,
  - minimum spanning trees,
  - maximum matchings,
  - scheduling problems,
  - ...

RSHs for Combinatorial Optimization

- Analysis of runtime and approximation quality on well-known combinatorial optimization problems, e. g.,
  - sorting problems (is this an optimization problem?),
  - covering problems,
  - cutting problems,
  - subsequence problems,
  - traveling salesperson problem,
  - Eulerian cycles,
  - shortest path problems,
  - minimum spanning trees,
  - maximum matchings,
  - scheduling problems,
  - ...
- We do not hope: to be better than the best problem-specific algorithms

RSHs for Combinatorial Optimization

- Analysis of runtime and approximation quality on well-known combinatorial optimization problems, e. g.,
  - sorting problems (is this an optimization problem?),
  - covering problems,
  - cutting problems,
  - subsequence problems,
  - traveling salesperson problem,
  - Eulerian cycles,
  - shortest path problems,
  - minimum spanning trees,
  - maximum matchings,
  - scheduling problems,
  - ...
- We do not hope: to be better than the best problem-specific algorithms
- In the following no fine-tuning of the results
- More details in the books (last slide)

Agenda

1. The origins: example functions and toy problems
   - A simple toy problem: OneMax for (1+1) EA
2. Combinatorial optimization problems
   - Minimum spanning trees
   - Maximum matchings
   - Shortest paths
   - Makespan scheduling
   - SA beats MA in combinatorial optimization
3. End

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

One bit for each edge.

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

One bit for each edge.

Decrease number of connected components, find minimum spanning tree:

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

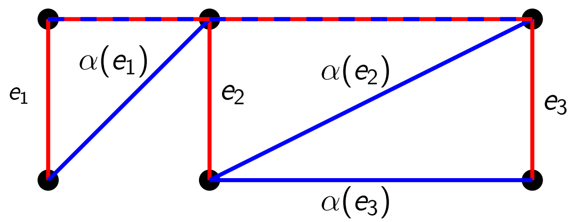
Minimization of f with respect to the lexicographic order.

Connected graph

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

Combinatorial Argument to Approach MSTs

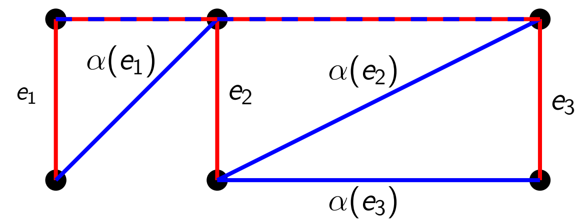
From arbitrary spanning tree T to MST T^* (Mayr/Plaxton, 1992):



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

Combinatorial Argument to Approach MSTs

From arbitrary spanning tree T to MST T^* (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

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; assume to be tree
- w_{opt} weight minimum spanning tree T^*

Upper Bound

Theorem

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; assume to be tree
- w_{opt} weight minimum spanning tree T^*

- Combinatorial argument $\to$ set of k operations to reach T^*
- (1+1) EA chooses operations uniformly
- $\Longrightarrow$ average weight decrease $(w(s) - w_{opt})/k$

Upper Bound

Concentrate on 2-bit flips:

- Expected weight decrease by a factor $1 - 1/k$ (or smaller $\rightarrow$ better) due to the good 2-bit flips
- Probability $\geq k/(em^2)$ for a good 2-bit flip
- Expected weight decrease $1 - 1/(em^2)$ in arbitrary step

Upper Bound

Concentrate on 2-bit flips:

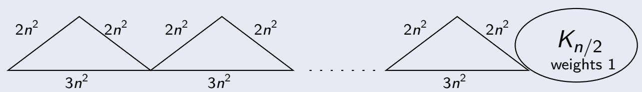
- Expected weight decrease by a factor $1 - 1/k$ (or smaller $\rightarrow$ better) due to the good 2-bit flips
- Probability $\geq k/(em^2)$ for a good 2-bit flip
- Expected weight decrease $1 - 1/(em^2)$ in arbitrary step

Method multiplicative drift drift analysis
(aka. expected multiplicative distance decrease):

- Have to bridge distance at most $D := w(s) - w_{\text{opt}} \leq m \cdot w_{\max}$.
- Relative improvement by factor $\delta := 1 - 1/(em^2)$
- Expected time $O((\ln D)/\delta) = O(m^2(\log n + \log w_{\max}))$

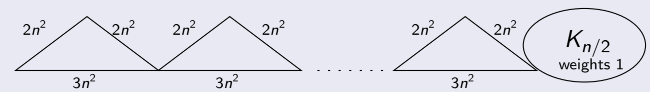
Further Results

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



Further Results

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



Related Results

- Experimental investigations
- Biased mutation operators
- $O(mn^2)$ for a multi-objective approach due to help objectives
- Approximations for multi-objective minimum spanning trees
- SA/MA and minimum spanning trees (Later!)

Agenda

Maximum Matchings

A matching in an undirected graph is a subset of pairwise disjoint edges;
aim: find a maximum matching (solvable in poly-time)

Maximum Matchings

A matching in an undirected graph is a subset of pairwise disjoint edges;
aim: find a maximum matching (solvable in poly-time)

Simple example: path of odd length

Maximum Matchings

A matching in an undirected graph is a subset of pairwise disjoint edges;
aim: find a maximum matching (solvable in poly-time)

Simple example: path of odd length

Maximum matching with more than half of edges

Maximum Matchings

A matching in an undirected graph is a subset of pairwise disjoint edges;
aim: find a maximum matching (solvable in poly-time)

Simple example: path of odd length

Suboptimal matching

Maximum Matchings

A matching in an undirected graph is a subset of pairwise disjoint edges;
aim: find a maximum matching (solvable in poly-time)

Simple example: path of odd length

Suboptimal matching

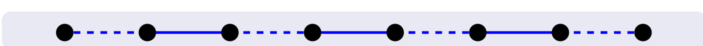
Concept: augmenting path

- Alternating between edges being inside and outside the matching
- Starting and ending at "free" nodes not incident on matching
- Flipping all choices along the path improves matching

Maximum Matchings

A matching in an undirected graph is a subset of pairwise disjoint edges;
aim: find a maximum matching (solvable in poly-time)

Simple example: path of odd length



Suboptimal matching

Concept: augmenting path

- Alternating between edges being inside and outside the matching
- Starting and ending at "free" nodes not incident on matching
- Flipping all choices along the path improves matching

Example: whole graph is augmenting path

Maximum Matchings

A matching in an undirected graph is a subset of pairwise disjoint edges;
aim: find a maximum matching (solvable in poly-time)

Simple example: path of odd length

Suboptimal matching

Concept: augmenting path

- Alternating between edges being inside and outside the matching
- Starting and ending at "free" nodes not incident on matching
- Flipping all choices along the path improves matching

Example: whole graph is augmenting path

Interesting: how simple EAs find augmenting paths

Maximum Matchings: Upper Bound

Fitness function $f\colon \{0,1\}^{\#\text{ edges}} \to \mathbb{R}$:

- one bit for each edge, value 1 iff edge chosen
- value for legal matchings: size of matching
- otherwise penalty leading to empty matching

Maximum Matchings: Upper Bound

Fitness function $f\colon \{0,1\}^{\#\text{ edges}} \to \mathbb{R}$:

- one bit for each edge, value 1 iff edge chosen
- value for legal matchings: size of matching
- otherwise penalty leading to empty matching

Example: path with $n+1$ nodes, n edges: bit string selects edges

Maximum Matchings: Upper Bound

Fitness function $f\colon \{0,1\}^{\#\text{ edges}} \to \mathbb{R}$:

- one bit for each edge, value 1 iff edge chosen
- value for legal matchings: size of matching
- otherwise penalty leading to empty matching

Example: path with $n+1$ nodes, n edges: bit string selects edges

Theorem

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

Maximum Matchings: Upper Bound (Ctnd.)

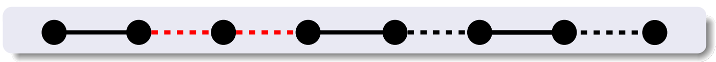
Proof idea for $O(n^4)$ bound

- Consider the level of second-best matchings.
- Fitness value does not change (walk on *plateau*).
- If "free" edge: chance to flip one bit! $\to$ probability $\Theta(1/n)$.
- Else steps flipping two bits $\to$ probability $\Theta(1/n^2)$.

Maximum Matchings: Upper Bound (Ctnd.)

Proof idea for $O(n^4)$ bound

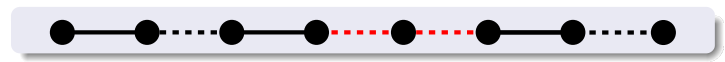
- Consider the level of second-best matchings.
- Fitness value does not change (walk on *plateau*).
- If "free" edge: chance to flip one bit! $\to$ probability $\Theta(1/n)$.
- Else steps flipping two bits $\to$ probability $\Theta(1/n^2)$.
- Shorten or lengthen augmenting path



Maximum Matchings: Upper Bound (Ctnd.)

Proof idea for $O(n^4)$ bound

- Consider the level of second-best matchings.
- Fitness value does not change (walk on *plateau*).
- If "free" edge: chance to flip one bit! $\to$ probability $\Theta(1/n)$.
- Else steps flipping two bits $\to$ probability $\Theta(1/n^2)$.
- Shorten or lengthen augmenting path

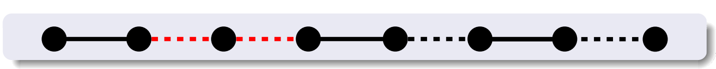


27/70
Carsten Witt Bioinspired Computation in Combinatorial Optimization

Maximum Matchings: Upper Bound (Ctnd.)

Proof idea for $O(n^4)$ bound

- Consider the level of second-best matchings.
- Fitness value does not change (walk on *plateau*).
- If "free" edge: chance to flip one bit! $\to$ probability $\Theta(1/n)$.
- Else steps flipping two bits $\to$ probability $\Theta(1/n^2)$.
- Shorten or lengthen augmenting path



27/70
Carsten Witt Bioinspired Computation in Combinatorial Optimization

Maximum Matchings: Upper Bound (Ctnd.)

Proof idea for $O(n^4)$ bound

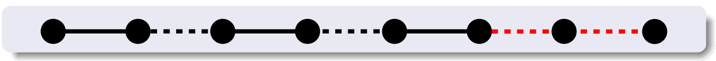
- Consider the level of second-best matchings.
- Fitness value does not change (walk on *plateau*).
- If "free" edge: chance to flip one bit! $\to$ probability $\Theta(1/n)$.
- Else steps flipping two bits $\to$ probability $\Theta(1/n^2)$.
- Shorten or lengthen augmenting path

27/70
Carsten Witt Bioinspired Computation in Combinatorial Optimization

Maximum Matchings: Upper Bound (Ctnd.)

Proof idea for $O(n^4)$ bound

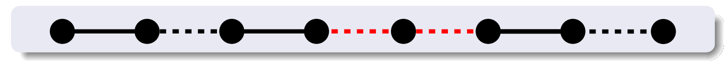
- Consider the level of second-best matchings.
- Fitness value does not change (walk on *plateau*).
- If "free" edge: chance to flip one bit! $\rightarrow$ probability $\Theta(1/n)$.
- Else steps flipping two bits $\rightarrow$ probability $\Theta(1/n^2)$.
- Shorten or lengthen augmenting path
- At length 1, chance to flip the free edge!



Maximum Matchings: Upper Bound (Ctnd.)

Proof idea for $O(n^4)$ bound

- Consider the level of second-best matchings.
- Fitness value does not change (walk on *plateau*).
- If "free" edge: chance to flip one bit! $\rightarrow$ probability $\Theta(1/n)$.
- Else steps flipping two bits $\rightarrow$ probability $\Theta(1/n^2)$.
- Shorten or lengthen augmenting path
- At length 1, chance to flip the free edge!



Maximum Matchings: Upper Bound (Ctnd.)

Proof idea for $O(n^4)$ bound

- Consider the level of second-best matchings.
- Fitness value does not change (walk on *plateau*).
- If "free" edge: chance to flip one bit! $\rightarrow$ probability $\Theta(1/n)$.
- Else steps flipping two bits $\rightarrow$ probability $\Theta(1/n^2)$.
- Shorten or lengthen augmenting path
- At length 1, chance to flip the free edge!

Maximum Matchings: Upper Bound (Ctnd.)

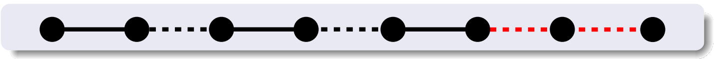
Proof idea for $O(n^4)$ bound

- Consider the level of second-best matchings.
- Fitness value does not change (walk on *plateau*).
- If "free" edge: chance to flip one bit! $\rightarrow$ probability $\Theta(1/n)$.
- Else steps flipping two bits $\rightarrow$ probability $\Theta(1/n^2)$.
- Shorten or lengthen augmenting path
- At length 1, chance to flip the free edge!

Maximum Matchings: Upper Bound (Ctnd.)

Proof idea for $O(n^4)$ bound

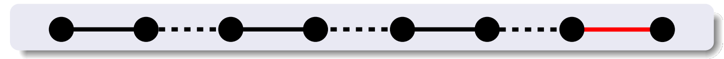
- Consider the level of second-best matchings.
- Fitness value does not change (walk on *plateau*).
- If “free” edge: chance to flip one bit! $\to$ probability $\Theta(1/n)$.
- Else steps flipping two bits $\to$ probability $\Theta(1/n^2)$.
- Shorten or lengthen augmenting path
- At length 1, chance to flip the free edge!



Maximum Matchings: Upper Bound (Ctnd.)

Proof idea for $O(n^4)$ bound

- Consider the level of second-best matchings.
- Fitness value does not change (walk on *plateau*).
- If “free” edge: chance to flip one bit! $\to$ probability $\Theta(1/n)$.
- Else steps flipping two bits $\to$ probability $\Theta(1/n^2)$.
- Shorten or lengthen augmenting path
- At length 1, chance to flip the free edge!



Maximum Matchings: Upper Bound (Ctnd.)

Proof idea for $O(n^4)$ bound

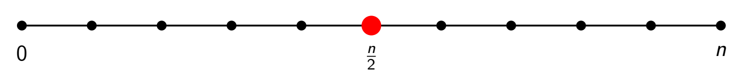
- Consider the level of second-best matchings.
- Fitness value does not change (walk on *plateau*).
- If “free” edge: chance to flip one bit! $\to$ probability $\Theta(1/n)$.
- Else steps flipping two bits $\to$ probability $\Theta(1/n^2)$.
- Shorten or lengthen augmenting path
- At length 1, chance to flip the free edge!

- Length changes according to a fair random walk
  $\to$ equal probability for lengthenings and shortenings

Fair Random Walk

Scenario: fair random walk

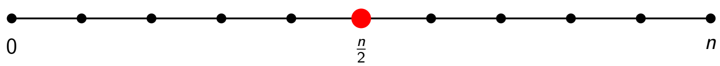
- Initially, player A and B both have $\frac{n}{2}$ \$
- Repeat: flip a coin
- If heads: A pays 1 \$ to B, tails: other way round
- Until one of the players is ruined.



Fair Random Walk

Scenario: fair random walk

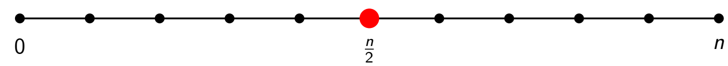
- Initially, player A and B both have $\frac{n}{2}$ \$
- Repeat: flip a coin
- If heads: A pays 1 \$ to B, tails: other way round
- Until one of the players is ruined.



Fair Random Walk

Scenario: fair random walk

- Initially, player A and B both have $\frac{n}{2}$ \$
- Repeat: flip a coin
- If heads: A pays 1 \$ to B, tails: other way round
- Until one of the players is ruined.

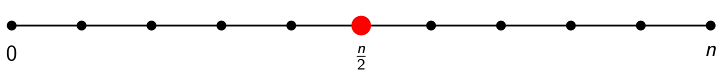


How long does the game take in expectation?

Fair Random Walk

Scenario: fair random walk

- Initially, player A and B both have $\frac{n}{2}$ \$
- Repeat: flip a coin
- If heads: A pays 1 \$ to B, tails: other way round
- Until one of the players is ruined.



How long does the game take in expectation?

Theorem:
Fair random walk on $\{0, \ldots, n\}$ takes in expectation $O(n^2)$ steps.

Maximum Matchings: Upper Bound (Ctnd.)

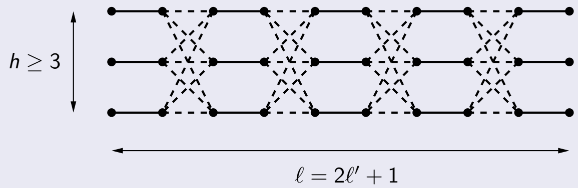
Proof idea for $O(n^4)$ bound

- Consider the level of second-best matchings.
- Fitness value does not change (walk on *plateau*).
- If “free” edge: chance to flip one bit! $\to$ probability $\Theta(1/n)$.
- Else steps flipping two bits $\to$ probability $\Theta(1/n^2)$.
- Shorten or lengthen augmenting path
- At length 1, chance to flip the free edge!

Length changes according to a fair random walk, expected $O(n^2)$ two-bit flips suffice, expected optimization time $O(n^2) \cdot O(n^2) = O(n^4)$.

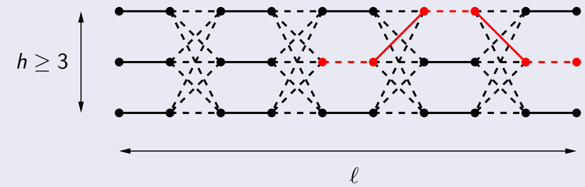
Maximum Matchings: Lower Bound

Worst-case graph $G_{h,\ell}$



Maximum Matchings: Lower Bound

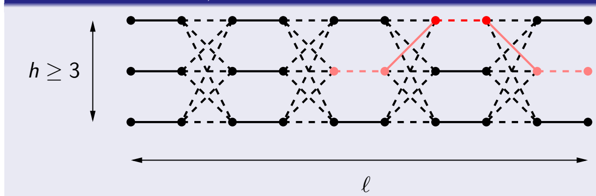
Worst-case graph $G_{h,\ell}$



Augmenting path

Maximum Matchings: Lower Bound

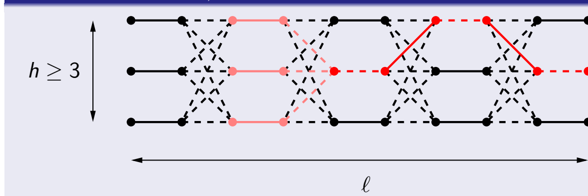
Worst-case graph $G_{h,\ell}$



Augmenting path can get shorter

Maximum Matchings: Lower Bound

Worst-case graph $G_{h,\ell}$



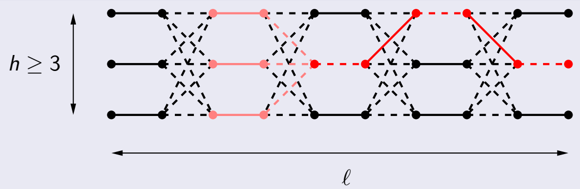
Augmenting path can get shorter but is more likely to get longer.
(unfair random walk)

Theorem

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

Maximum Matchings: Lower Bound

Worst-case graph $G_{h,\ell}$



Augmenting path can get shorter but is more likely to get longer.
(unfair random walk)

Theorem

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

Proof requires analysis of negative drift (simplified drift theorem).

Maximum Matching: Approximations

Insight: do not hope for exact solutions but for approximations

For maximization problems: solution with value a is called $(1+\varepsilon)$-approximation if $\frac{\mathrm{OPT}}{a} \leq 1+\varepsilon$, where OPT optimal value.

Maximum Matching: Approximations

Insight: do not hope for exact solutions but for approximations

For maximization problems: solution with value a is called $(1+\varepsilon)$-approximation if $\frac{\mathrm{OPT}}{a} \leq 1+\varepsilon$, where OPT optimal value.

Theorem

For $\varepsilon > 0$, (1+1) EA finds a $(1+\varepsilon)$-approximation of a maximum matching in expected time $O(m^{2/\varepsilon+2})$ (m number of edges).

Maximum Matching: Approximations

Insight: do not hope for exact solutions but for approximations

For maximization problems: solution with value a is called $(1+\varepsilon)$-approximation if $\frac{\mathrm{OPT}}{a} \leq 1+\varepsilon$, where OPT optimal value.

Theorem

For $\varepsilon > 0$, (1+1) EA finds a $(1+\varepsilon)$-approximation of a maximum matching in expected time $O(m^{2/\varepsilon+2})$ (m number of edges).

Proof idea: If current solution worse than $(1+\varepsilon)$-approximate, there is a "short" augmenting path (length $\leq 2/\varepsilon+1$); flip it in one go.

Agenda

All-pairs-shortest-path (APSP) problem

Given: Connected directed graph $G = (V, E)$, $|V| = n$ and $|E| = m$, and a function $w: E \to N$ which assigns positive integer weights to the edges.

Compute from each vertex $v_i \in V$ a shortest path (path of minimal weight) to every other vertex $v_j \in V \setminus \{v_i\}$

Representation:

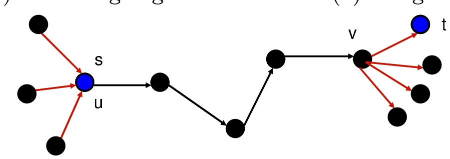
Individuals are paths between two particular vertices v_i and v_j

Initial Population: $P := \{I_{u,v} = (u, v) | (u, v) \in E\}$

Mutation:

Pick individual $I_{u,v}$ uniformly at random

$E^-(u)$: incoming edges of u $\quad$ $E^+(v)$: outgoing edges of v



Pick uniformly at random an edge $e = (x, y) \in E^-(u) \cup E^+(v)$

Add e $\quad$ New individual $I'_{s,t}$

Mutation-based EA

Steady State EA

1. Set $P = \{I_{u,v} = (u,v) \mid (u,v) \in E\}$.
2. Choose an individual $I_{x,y} \in P$ uniformly at random.
3. Mutate $I_{x,y}$ to obtain an individual $I'_{s,t}$.
4. If there is no individual $I_{s,t} \in P$, $P = P \cup \{I'_{s,t}\}$, else if $f(I'_{s,t}) \leq f(I_{s,t})$, $P = (P \cup \{I'_{s,t}\}) \setminus \{I_{s,t}\}$
5. Repeat Steps 2–4 forever.

Lemma:

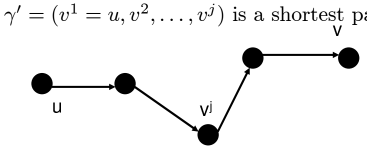
Let $\ell \geq \log n$. The expected time until has found all shortest paths with at most ℓ edges is $O(n^3 \ell)$.

Proof idea:

Consider two vertices u and v, $u \neq v$.

Let $\gamma := (v^1 = u, v^2, \ldots, v^{\ell'+1} = v)$ be a shortest path from u to v consisting of ℓ', $\ell' \leq \ell$, edges in G

the sub-path $\gamma' = (v^1 = u, v^2, \ldots, v^j)$ is a shortest path from u to v^j.



Population size is upper bounded n^2
(for each pair of vertices at most one path)

- Pick shortest path from u to v_j and append edge (v_j, v_{j+1})
- Shortest path from u to v_{j+1}

- Probability to pick $I_{u,vj}$ is at least $1/n^2$
- Probability to append right edge is at least $1/(2n)$
- Success with probability at least $p = 1/(2n^3)$
- At most l successes needed to obtain shortest path from u to v

Consider typical run consisting of $T=cn^3l$ steps.

What is the probability that the shortest path from u to v has been obtained?

We need at most l successes, where a success happens in each step with probability at least $p = 1/(2n^3)$

Define for each step i a random variable X_i.

$X_i = 1$ if step i is a success

$X_i = 0$ if step i is not a success

Analysis

$Prob(X_i = 1) \geq p = 1/(2n^3) \quad X = \sum_{i=1}^{T} X_i \qquad X \geq \ell$???

Expected number of successes $E(X) \geq T/(2n^3) = \frac{cn^3\ell}{2n^3} = \frac{c\ell}{2}$

Chernoff: $Prob(X < (1-\delta)E(x)) \leq e^{-E(X)\delta^2/2}$

$\delta = \frac{1}{2}$

$Prob(X < (1-\frac{1}{2})E(x)) \leq e^{-E(X)/8} \leq e^{-T/(16n^3)} = e^{-cn^3\ell/(16n^3)} = e^{-c\ell/(16)}$

Probability for failure of at least one pair of vertices at most: $n^2 \cdot e^{-c\ell/16}$

c large enough and $\ell \geq \log n$:

No failure in any path with probability at least $\alpha = 1 - n^2 \cdot e^{-c\ell/16} = 1 - o(1)$

Holds for any phase of T steps

Expected time upper bound by $T/\alpha = O(n^3\ell)$

Shortest paths have length at most n-1.

Set l = n-1

Theorem

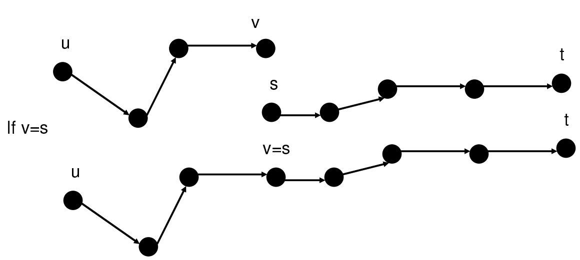
The expected optimization time of Steady State EA for the APSP problem is $O(n^4)$.

Remark:
There are instances where the expected optimization of $(\mu+1)$-EA is $\Omega(n^4)$

Question:
Can crossover help to achieve a better expected optimization time?

Crossover

Pick two individuals $I_{u,v}$ and $I_{s,t}$ from population uniformly at random.



Steady State GA

1. Set $P = \{I_{u,v} = (u,v) \mid (u,v) \in E\}$.
2. Choose $r \in [0,1]$ uniformly at random.
3. If $r \leq p_c$, choose two individuals $I_{x,y} \in P$ and $I_{x',y'} \in P$ uniformly at random and perform crossover to obtain an individual $I'_{s,t}$,
   else choose an individual $I_{x,y} \in P$ uniformly at random and mutate $I_{x,y}$ to obtain an individual $I'_{s,t}$.
4. If $I'_{s,t}$ is a path from s to t then
   - ★ If there is no individual $I_{s,t} \in P$, $P = P \cup \{I'_{s,t}\}$,
   - ★ else if $f(I'_{s,t}) \leq f(I_{s,t})$, $P = (P \cup \{I'_{s,t}\}) \setminus \{I_{s,t}\}$.
5. Repeat Steps 2–4 forever.

p_c is a constant

Theorem:
The expected optimization time of Steady State GA is $O(n^{3.5}\sqrt{\log n})$.

Mutation and $\ell^* := \sqrt{n \log n}$

All shortest path of length at most l* edges are obtained

Show: Longer paths are obtained by crossover within the stated time bound.

Analysis Crossover

Long paths by crossover:

Assumption: All shortest paths with at most l* edges have already been obtained.

Assume that all shortest paths of length k ≤ l* have been obtained.

What is the expected time to obtain all shortest paths of length at most 3k/2?

Analysis Crossover

Consider pair of vertices x and y for which a shortest path of r, k < r ≤ 3k/2, edges exists.

There are 2k-r pairs of shortest paths of length at most k that can be joined to obtain shortest path from x to y.

Probability for one specific pair: at least $1/n^4$

At least 2k+1-r possible pairs: probability at least $(2k+1-r)/n^4) \geq k/(2n^4)$

At most n^2 shortest paths of length r, k < r ≤ 3k/2

Time to collect all paths $O(n^4 \log n/ k)$
(similar to Coupon Collectors Theorem)

Analysis Crossover

Sum up over the different values of k, namely

$$\sqrt{n \log n}, c \cdot \sqrt{n \log n}, c^2 \cdot \sqrt{n \log n}, \ldots, c^{\log_c(n/\sqrt{n \log n})} \cdot \sqrt{n \log n},$$

where $c = 3/2$.

Expected Optimization

$$\sum_{s=0}^{\log_c(n/\sqrt{n \log n})} \left(O\left(\frac{n^4 \log n}{\sqrt{n \log n}}\right) c^{-s} \right) = O(n^{3.5}\sqrt{\log n}) \sum_{s=0}^{\infty} c^{-s} = O(n^{3.5}\sqrt{\log n})$$

Agenda

1. The origins: example functions and toy problems
   - A simple toy problem: OneMax for (1+1) EA
2. Combinatorial optimization problems
   - Minimum spanning trees
   - Maximum matchings
   - Shortest paths
   - Makespan scheduling
   - SA beats MA in combinatorial optimization
3. End

Makespan Scheduling

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

Makespan Scheduling

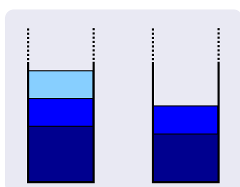
What about NP-hard problems? $\rightarrow$ Study approximation quality

Makespan scheduling on 2 machines:

- n objects with weights/processing times $w_1, \ldots, w_n$
- 2 machines (bins)
- Minimize the total weight of fuller bin = makespan.

Formally, find $I \subseteq \{1, \ldots, n\}$ minimizing

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



Makespan Scheduling

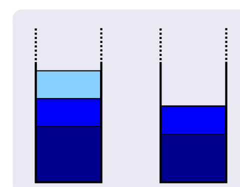
What about NP-hard problems? $\rightarrow$ Study approximation quality

Makespan scheduling on 2 machines:

- n objects with weights/processing times $w_1, \ldots, w_n$
- 2 machines (bins)
- Minimize the total weight of fuller bin = makespan.

Formally, find $I \subseteq \{1, \ldots, n\}$ minimizing

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



Sometimes also called the Partition problem.
This is an "easy" NP-hard problem, good approximations possible

Fitness Function

- Problem encoding: bit string $x_1, \dots, x_n$ reserves a bit for each object, put object i in bin $x_i + 1$.
- Fitness function

$$f(x_1, \dots, x_n) := \max\left\{\sum_{i=1}^{n} w_i x_i, \sum_{i=1}^{n} w_i(1 - x_i)\right\}$$

  to be minimized.
- Consider (1+1) EA and RLS.

Types of Results

- Worst-case results
- Success probabilities and approximations
- An average-case analysis

Sufficient Conditions for Progress

Abbreviate $S := w_1 + \cdots + w_n \Rightarrow$ perfect partition has cost $\frac{S}{2}$.

Sufficient Conditions for Progress

Abbreviate $S := w_1 + \cdots + w_n \Rightarrow$ perfect partition has cost $\frac{S}{2}$.

Suppose we know

- s^* = size of smallest object in the fuller bin,

Sufficient Conditions for Progress

Abbreviate $S := w_1 + \cdots + w_n \Rightarrow$ perfect partition has cost $\frac{S}{2}$.

Suppose we know

- s^* = size of smallest object in the fuller bin,
- $f(x) > \frac{S}{2} + \frac{s^*}{2}$ for the current search point x

then the solution is improvable by a single-bit flip.

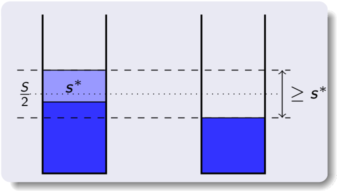
Sufficient Conditions for Progress

Abbreviate $S := w_1 + \cdots + w_n \Rightarrow$ perfect partition has cost $\frac{S}{2}$.

Suppose we know

- s^* = size of smallest object in the fuller bin,
- $f(x) > \frac{S}{2} + \frac{s^*}{2}$ for the current search point x

then the solution is improvable by a single-bit flip.



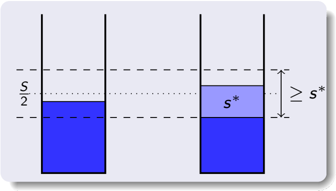
Sufficient Conditions for Progress

Abbreviate $S := w_1 + \cdots + w_n \Rightarrow$ perfect partition has cost $\frac{S}{2}$.

Suppose we know

- s^* = size of smallest object in the fuller bin,
- $f(x) > \frac{S}{2} + \frac{s^*}{2}$ for the current search point x

then the solution is improvable by a single-bit flip.



If $f(x) < \frac{S}{2} + \frac{s^*}{2}$, no improvements can be guaranteed.

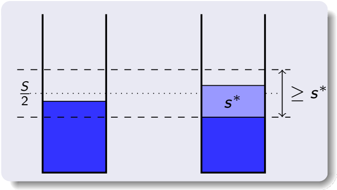
Sufficient Conditions for Progress

Abbreviate $S := w_1 + \cdots + w_n \Rightarrow$ perfect partition has cost $\frac{S}{2}$.

Suppose we know

- s^* = size of smallest object in the fuller bin,
- $f(x) > \frac{S}{2} + \frac{s^*}{2}$ for the current search point x

then the solution is improvable by a single-bit flip.



If $f(x) < \frac{S}{2} + \frac{s^*}{2}$, no improvements can be guaranteed.

Lemma

If smallest object in fuller bin is always bounded by s^* then (1+1) EA and RLS reach f-value $\leq \frac{S}{2} + \frac{s^*}{2}$ in expected $O(n^2)$ steps.

Worst-Case Results

Theorem

On any instance to the makespan scheduling problem, the (1+1) EA and RLS reach a solution with approximation ratio $\frac{4}{3}$ in expected time $O(n^2)$.

Use study of object sizes and previous lemma.

Worst-Case Results

Theorem

On any instance to the makespan scheduling problem, the (1+1) EA and RLS reach a solution with approximation ratio $\frac{4}{3}$ in expected time $O(n^2)$.

Use study of object sizes and previous lemma.

Theorem

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

Worst-Case Instance

Instance $W_\varepsilon^* = \{w_1, \dots, w_n\}$ is defined by $w_1 := w_2 := \frac{1}{3} - \frac{\varepsilon}{4}$ (big objects) and $w_i := \frac{1/3+\varepsilon/2}{n-2}$ for $3 \le i \le n$, ε very small constant; n even

Worst-Case Instance

Instance $W_\varepsilon^* = \{w_1, \dots, w_n\}$ is defined by $w_1 := w_2 := \frac{1}{3} - \frac{\varepsilon}{4}$ (big objects) and $w_i := \frac{1/3+\varepsilon/2}{n-2}$ for $3 \le i \le n$, ε very small constant; n even

Sum is 1; there is a perfect partition.

Worst-Case Instance

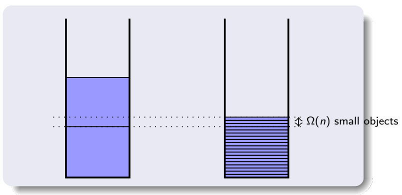
Instance $W^*_\varepsilon = \{w_1, \dots, w_n\}$ is defined by $w_1 := w_2 := \frac{1}{3} - \frac{\varepsilon}{4}$ (big objects) and $w_i := \frac{1/3+\varepsilon/2}{n-2}$ for $3 \le i \le n$, ε very small constant; n even

Sum is 1; there is a perfect partition.

But if one bin with big and one bin with small objects: value $\frac{2}{3} - \frac{\varepsilon}{2}$.

Move a big object in the emptier bin $\Rightarrow$ value $(\frac{1}{3} + \frac{\varepsilon}{2}) + (\frac{1}{3} - \frac{\varepsilon}{4}) = \frac{2}{3} + \frac{\varepsilon}{4}$!

Need to move $\ge \varepsilon n$ small objects at once for improvement: very unlikely.



With constant probability in this situation, $n^{\Omega(n)}$ needed to escape.

Worst Case – PRAS by Parallelism

Previous result shows: success dependent on big objects

Theorem

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

Worst Case – PRAS by Parallelism

Previous result shows: success dependent on big objects

Theorem

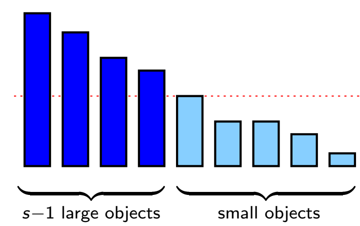
On any instance, the (1+1) EA and RLS with prob. $\ge 2^{-c\lceil 1/\varepsilon\rceil \ln(1/\varepsilon)}$ find a $(1+\varepsilon)$-approximation within $O(n \ln(1/\varepsilon))$ steps.

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

Worst Case – PRAS by Parallelism (Proof Idea)

Set $s := \lceil \frac{2}{\varepsilon} \rceil$

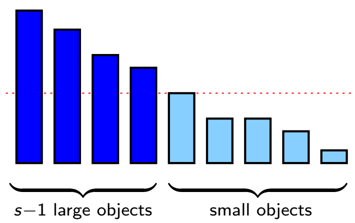
Assuming $w_1 \ge \dots \ge w_n$, we have $w_i \le \varepsilon \frac{S}{2}$ for $i \ge s$.



Worst Case – PRAS by Parallelism (Proof Idea)

Set $s := \lceil \frac{2}{\varepsilon} \rceil$

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



analyze probability of distributing

- large objects in an optimal way,
- small objects greedily $\Rightarrow$ error $\leq \varepsilon \frac{S}{2}$,

Random search rediscovers algorithmic idea of early algorithms.

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.

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?

Makespan Scheduling – Known Averge-Case Results

Deterministic, problem-specific heuristic LPT

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

Known for both random models:

LPT creates a solution with discrepancy $O((\log n)/n)$.

Makespan Scheduling – Known Averge-Case Results

Deterministic, problem-specific heuristic LPT

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

Known for both random models:
LPT creates a solution with discrepancy $O((\log n)/n)$.

What discrepancy do the (1+1) EA and RLS reach in poly-time?

Average-Case Analysis of the (1+1) EA

Theorem

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!

Average-Case Analysis of the (1+1) EA

Theorem

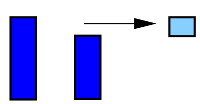
In both models, the (1+1) EA reaches discrepancy $O((\log n)/n)$ after $O(n^{c+4}\log^2 n)$ steps with probability $1 - O(1/n^c)$.

Almost the same result as for LPT!

Proof exploits order statistics:

If $X_{(i)}$ (i-th largest) in fuller bin, $X_{(i+1)}$ in emptier one, and discrepancy $> 2(X_{(i)} - X_{(i+1)}) > 0$, then objects can be swapped; discrepancy falls

Consider such "difference objects".



Average-Case Analysis of the (1+1) EA

Theorem

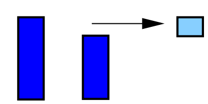
In both models, the (1+1) EA reaches discrepancy $O((\log n)/n)$ after $O(n^{c+4}\log^2 n)$ steps with probability $1 - O(1/n^c)$.

Almost the same result as for LPT!

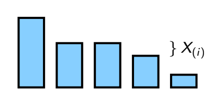
Proof exploits order statistics:

If $X_{(i)}$ (i-th largest) in fuller bin, $X_{(i+1)}$ in emptier one, and discrepancy $> 2(X_{(i)} - X_{(i+1)}) > 0$, then objects can be swapped; discrepancy falls

Consider such "difference objects".



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



Agenda

1. The origins: example functions and toy problems
   - A simple toy problem: OneMax for (1+1) EA
2. Combinatorial optimization problems
   - Minimum spanning trees
   - Maximum matchings
   - Shortest paths
   - Makespan scheduling
   - SA beats MA in combinatorial optimization
3. End

Simulated Annealing vs. Metropolis

Metropolis Algorithm (MA) and Simulated Annealing (SA)
for the minimization of functions $f\colon \{0,1\}^n \to \mathbb{R}$

1. $t := 0$. Choose $x \in \{0,1\}^n$ uniformly at random.
2. $y := x$.
3. Flip exactly one bit in y chosen uniformly at random.
4. If $f(y) \leq f(x)$ then $x := y$ else $x := y$ with probability $e^{\frac{f(x)-f(y)}{T_t}}$
5. $t := t+1$. Continue at line 2.

Simulated Annealing vs. Metropolis

Metropolis Algorithm (MA) and Simulated Annealing (SA)
for the minimization of functions $f\colon \{0,1\}^n \to \mathbb{R}$

1. $t := 0$. Choose $x \in \{0,1\}^n$ uniformly at random.
2. $y := x$.
3. Flip exactly one bit in y chosen uniformly at random.
4. If $f(y) \leq f(x)$ then $x := y$ else $x := y$ with probability $e^{\frac{f(x)-f(y)}{T_t}}$
5. $t := t+1$. Continue at line 2.

Simulated Annealing vs. Metropolis

Metropolis Algorithm (MA) and Simulated Annealing (SA)
for the minimization of functions $f\colon \{0,1\}^n \to \mathbb{R}$

1. $t := 0$. Choose $x \in \{0,1\}^n$ uniformly at random.
2. $y := x$.
3. Flip exactly one bit in y chosen uniformly at random.
4. If $f(y) \leq f(x)$ then $x := y$ else $x := y$ with probability $e^{\frac{f(x)-f(y)}{T_t}}$
5. $t := t+1$. Continue at line 2.

Given positive T_t, worse search point can be accepted and escapes from local optima (where RLS would be stuck) are possible.

Probability of accepting a worsening depends on T_t and fitness difference.

Simulated Annealing vs. Metropolis

Metropolis Algorithm (MA) and Simulated Annealing (SA)
for the minimization of functions $f\colon \{0,1\}^n \to \mathbb{R}$

1. $t := 0$. Choose $x \in \{0,1\}^n$ uniformly at random.
2. $y := x$.
3. Flip exactly one bit in y chosen uniformly at random.
4. If $f(y) \le f(x)$ then $x := y$ else $x := y$ with probability $e^{\frac{f(x)-f(y)}{T_t}}$
5. $t := t+1$. Continue at line 2.

Given positive T_t, worse search point can be accepted and escapes from local optima (where RLS would be stuck) are possible.

Probability of accepting a worsening depends on T_t and fitness difference.

Example: With $T = 1$, f-increase by 1 accepted w. prob. $e^{-1} = \Omega(1)$.

61/70

Carsten Witt Bioinspired Computation in Combinatorial Optimization

Simulated Annealing vs. Metropolis

Metropolis Algorithm (MA) and Simulated Annealing (SA)
for the minimization of functions $f\colon \{0,1\}^n \to \mathbb{R}$

1. $t := 0$. Choose $x \in \{0,1\}^n$ uniformly at random.
2. $y := x$.
3. Flip exactly one bit in y chosen uniformly at random.
4. If $f(y) \le f(x)$ then $x := y$ else $x := y$ with probability $e^{\frac{f(x)-f(y)}{T_t}}$
5. $t := t+1$. Continue at line 2.

Given positive T_t, worse search point can be accepted and escapes from local optima (where RLS would be stuck) are possible.

Probability of accepting a worsening depends on T_t and fitness difference.

Example 2: With $T = 0$, f-increase accepted w. prob. $e^{-\infty} = 0$.

61/70

Carsten Witt Bioinspired Computation in Combinatorial Optimization

Simulated Annealing vs. Metropolis

Metropolis Algorithm (MA) and Simulated Annealing (SA)
for the minimization of functions $f\colon \{0,1\}^n \to \mathbb{R}$

1. $t := 0$. Choose $x \in \{0,1\}^n$ uniformly at random.
2. $y := x$.
3. Flip exactly one bit in y chosen uniformly at random.
4. If $f(y) \le f(x)$ then $x := y$ else $x := y$ with probability $e^{\frac{f(x)-f(y)}{T_t}}$
5. $t := t+1$. Continue at line 2.

Typical distinction:
T_t fixed, i. e., independent of t $\to$ heuristic is called MA.
T_t varies depending on t $\to$ heuristic is called SA.

61/70

Carsten Witt Bioinspired Computation in Combinatorial Optimization

Simulated Annealing Beats Metropolis in Combinatorial Optimization

SA's choice of T_t is usually called cooling schedule.

62/70

Carsten Witt Bioinspired Computation in Combinatorial Optimization

Simulated Annealing Beats Metropolis in Combinatorial Optimization

SA's choice of T_t is usually called cooling schedule.

Typically: T_t decreases (cools down) with t to "simulate annealing"
$\rightarrow$ large jumps in the beginning, later only small changes

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 Optimization

SA's choice of T_t is usually called cooling schedule.

Typically: T_t decreases (cools down) with t to "simulate annealing"
$\rightarrow$ large jumps in the beginning, later only small changes

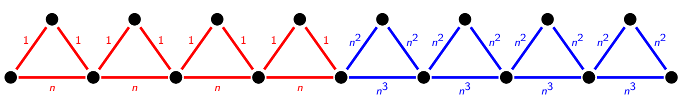
Jerrum/Sinclair (1996)

"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.

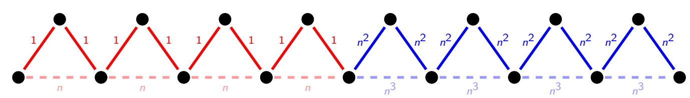
The MST Instance: Connected Triangles

Let $n = 6k$. Instance consists of $n/3$ connected triangles, half light and half heavy. Each triangle has two light and one heavy edge.



The MST Instance: Connected Triangles

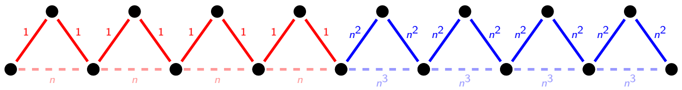
Let $n = 6k$. Instance consists of $n/3$ connected triangles, half light and half heavy. Each triangle has two light and one heavy edge.



Optimal solution is unique and chooses only light edges.

The MST Instance: Connected Triangles

Let $n = 6k$. Instance consists of $n/3$ connected triangles, half light and half heavy. Each triangle has two light and one heavy edge.



Optimal solution is unique and chooses only light edges.

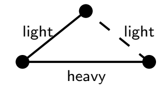
Claims: MA with arbitrary temperature typically needs exponential time on the connected-triangles instance (= inefficient). SA with an appropriate cooling schedule typically finds optimum in polynomial time (= efficient).

→ "SA Beats Metropolis in Combinatorial Optimization"

Proof idea: need different temperatures to optimize all triangles.

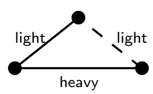
Proof Idea

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



Proof Idea

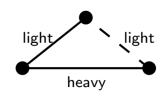
Concentrate on wrong triangles:
one heavy, one light edge chosen



- 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.

Proof Idea

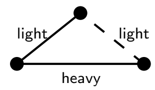
Concentrate on wrong triangles:
one heavy, one light edge chosen



- 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.
- Such flip leads to a worse spanning tree
  → need high temperature T^* to correct wrong heavy triangles.

Proof Idea

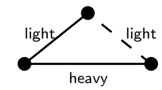
Concentrate on wrong triangles:
one heavy, one light edge chosen



- 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.
- 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.

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.
- 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.

The Traveling Salesperson Problem

MST is a polynomial-time solvable problem.

The Traveling Salesperson Problem

MST is a polynomial-time solvable problem.

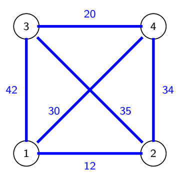
Can SA beat MA also on an NP-hard problem (i. e., problems for which no optimal polynomial-time algorithm is known)?

The Traveling Salesperson Problem

MST is a polynomial-time solvable problem.

Can SA beat MA also on an NP-hard problem (i. e., problems for which no optimal polynomial-time algorithm is known)?

Traveling Salesperson Problem (TSP), a notorious NP-hard problem: Given a complete graph on the vertex set $V = \{1, \ldots, n\}$ and edge costs $c(i,j) \in \mathbb{R}^+$, find a permutation $\pi \in S_n$ resulting in a Hamiltonian circuit (round trip) of minimum total cost.

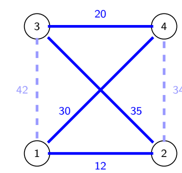


The Traveling Salesperson Problem

MST is a polynomial-time solvable problem.

Can SA beat MA also on an NP-hard problem (i. e., problems for which no optimal polynomial-time algorithm is known)?

Traveling Salesperson Problem (TSP), a notorious NP-hard problem: Given a complete graph on the vertex set $V = \{1, \ldots, n\}$ and edge costs $c(i,j) \in \mathbb{R}^+$, find a permutation $\pi \in S_n$ resulting in a Hamiltonian circuit (round trip) of minimum total cost.



SA/MA for the TSP

Search space: S_n (all permuations on $\{1, \ldots, n\}$)

Initialization: tour $1, \ldots, n$

SA/MA for the TSP

Search space: S_n (all permuations on $\{1, \ldots, n\}$)

Initialization: tour $1, \ldots, n$

What mutation operator to use ("how to flip a bit")?

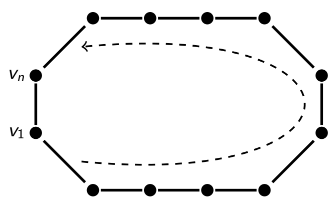
SA/MA for the TSP

Search space: S_n (all permuations on $\{1, \dots, n\}$)

Initialization: tour $1, \dots, n$

What mutation operator to use ("how to flip a bit")?

Solution: 2-opt local change



1. Let $v_1, \dots, v_n$ be the vertices on the current TSP tour.

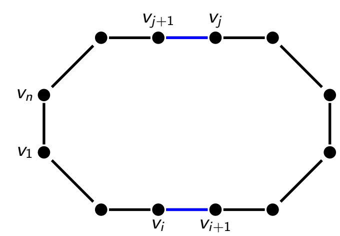
SA/MA for the TSP

Search space: S_n (all permuations on $\{1, \dots, n\}$)

Initialization: tour $1, \dots, n$

What mutation operator to use ("how to flip a bit")?

Solution: 2-opt local change



1. Let $v_1, \dots, v_n$ be the vertices on the current TSP tour.
2. Choose two edges (v_i, v_{i+1}) and (v_j, v_{j+1}), $j > i$, uniformly.

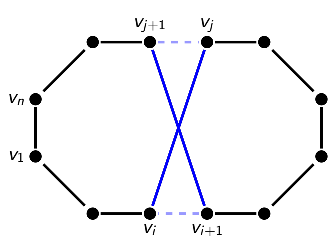
SA/MA for the TSP

Search space: S_n (all permuations on $\{1, \dots, n\}$)

Initialization: tour $1, \dots, n$

What mutation operator to use ("how to flip a bit")?

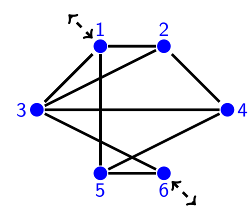
Solution: 2-opt local change



1. Let $v_1, \dots, v_n$ be the vertices on the current TSP tour.
2. Choose two edges (v_i, v_{i+1}) and (v_j, v_{j+1}), $j > i$, uniformly.
3. Let the new tour be $(v_1, \dots, v_i) + (v_j, v_{j-1}, \dots, v_{i+1}) + (v_{j+1}, \dots, v_n)$.

The TSP Instance – Skeletons

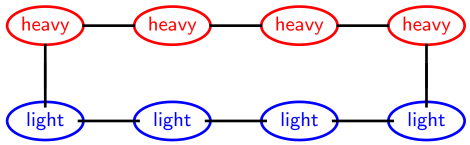
Role of former triangles now taken by Skeleton Graph.



- Implicit entries/exits 1 and 6
- $(1, 3)$ will become heaviest edge, $(1, 2)$ second-heaviest; all other edges light.
- Three possible paths visiting all vertices: $p_{\mathrm{wst}} = 132456$, $p_{\mathrm{mid}} = 123456$, $p_{\mathrm{opt}} = 154236$ (listed according to falling cost)
- SA starts with p_{mid}.
- Only possible transitions by 2-opt: $p_{\mathrm{mid}} \leftrightarrow p_{\mathrm{wst}} \leftrightarrow p_{\mathrm{opt}}$
- No direct transition from p_{mid} to p_{opt} $\Rightarrow$ intermediate worsening necessary
- Used in a heavy and a light variant

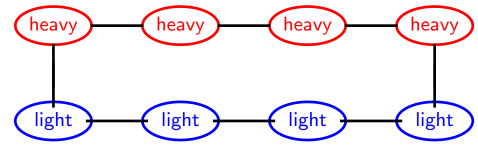
The TSP Instance – Results

Final TSP instance is composed of equally many light and heavy skeletons.



The TSP Instance – Results

Final TSP instance is composed of equally many light and heavy skeletons.



All skeletons must be corrected at least once. Then:

- MA with fixed temperature fails either at the heavy or light skeletons.
- SA with appropriate cooling schedule will optimize heavy and light skeletons one after another.

Summary and Conclusions

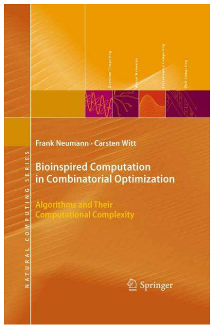
- Analysis of RSHs in combinatorial optimization
- Starting from toy problems to real problems
- Surprising results
- Interesting techniques
- Analysis of new approaches possible (→ other GECCO tutorials)

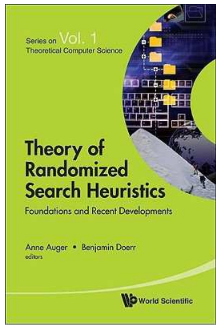
Summary and Conclusions

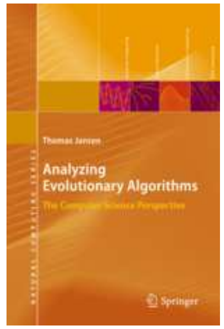
- Analysis of RSHs in combinatorial optimization
- Starting from toy problems to real problems
- Surprising results
- Interesting techniques
- Analysis of new approaches possible (→ other GECCO tutorials)

→ An exciting research direction.

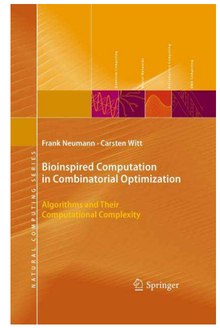
Suggested Reading

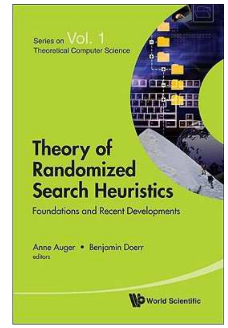


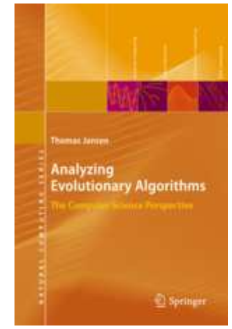




Suggested Reading







Thank you!