Stochastic Local Search Heuristics

Thorsten Koch

Setting the stage

"Combinatorial Optimization searches for an optimum object in a finite collection of objects. Typically, the collection has a concise representation, while the number of objects is huge --- more precisely, grows exponentially in the size of the representation. So scanning all objects one by one and selecting the best one is not an option."

— Alexander Schrijver, Combinatorial Optimization, 2003, Page 1.

$$\min_{\boldsymbol{x} \in \boldsymbol{X}} \boldsymbol{f}(\boldsymbol{x}) \text{ with } \boldsymbol{X} = \left\{ \boldsymbol{x}, \boldsymbol{b}, \underline{\boldsymbol{l}}, \overline{\boldsymbol{u}} \in \mathbb{Z}^{\boldsymbol{n}} : \boldsymbol{g}(\boldsymbol{x}) \leq \boldsymbol{b}, \underline{\boldsymbol{l}} \leq \boldsymbol{x} \leq \overline{\boldsymbol{u}} \right\}$$

For the rest of the talk, we assume: $f: X \to \mathbb{Z}$ is a linear or quadratic function, i.e., $f(x) = c^T x + x^T Q x, c \in \mathbb{Z}^n, Q \in \mathbb{Z}^{n \times n}$, and $g\text{:}\ X \to \mathbb{Z}^n$ is a linear function, i.e., $\mathrm{g}(x) = Ax, A \in \mathbb{Z}^{n \times n}$.

Note: $\operatorname{argmin} f(x) = \operatorname{argmax} -f(x)$ and $g(x) + s = b, s \geq 0 \iff g(x) \leq 0$, and similar for $\geq$.

We defined everything using integer numbers. If we would use rational numbers, we could then scale them by the least common multiple of all denominators to make everything integer.

Examples for finding some $x \in X$ is difficult. Might be, but doesn't have to.

We can write our problem as a decision problem (and minimize by binary search):

$$\boldsymbol{X_k} \neq \emptyset \textbf{ ? } \text{with } \boldsymbol{X_k} = \left\{\boldsymbol{x} \in \mathbb{Z}^{\boldsymbol{n}} : \boldsymbol{Ax} \leq \boldsymbol{b}, \underline{\boldsymbol{l}} \leq \boldsymbol{x} \leq \overline{\boldsymbol{u}} \ \wedge \boldsymbol{c}^{\boldsymbol{T}}\boldsymbol{x} = \boldsymbol{k}\right\}$$

In this case finding some x is equivalent to solving the problem.

Or, using some suitable big constant M, we can move the constraints into the objective:

$$\min_{\boldsymbol{x} \in \boldsymbol{X}} \boldsymbol{f}(\boldsymbol{x}) \text{ with } \boldsymbol{X} = \left\{\boldsymbol{x} \in \mathbb{Z}^{\boldsymbol{n}} : \underline{\boldsymbol{l}} \leq \boldsymbol{x} \leq \overline{\boldsymbol{u}}\right\}, \boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{c}^{\boldsymbol{T}}\boldsymbol{x} + \boldsymbol{M}(\boldsymbol{b} - \boldsymbol{Ax})^2$$

now it is obviously trivial to find some $x \in X$.

This difference in difficulty is one reason why people believe $P \neq NP$

Note: To solve an ILP, i.e., to optimality two things must be done:

(1) $\exists$:Find the minimum $x^* \in X$.

(2) $\forall$:Prove there exists no $x^* \in X$ with $f(x) < f(x^*)$.

This is equivalent to showing: $\left\{\boldsymbol{x} \in \mathbb{Z}^{\boldsymbol{n}} : \boldsymbol{Ax} \leq \boldsymbol{b}, \underline{\boldsymbol{l}} \leq \boldsymbol{x} \leq \overline{\boldsymbol{u}} \ \wedge \boldsymbol{f}(\boldsymbol{x}) < \boldsymbol{f}(\boldsymbol{x}^*)\right\} = \emptyset$

What does “solving” an NP-hard problem typically mean?

	Being able to …
Theoretical Computer Scientist	… compute **proven optimal solutions to every instance** of this problem class with at most this effort
Applied Discrete Mathematician	… practically compute within numerical tolerances **proven optimal solutions to these particular (relevant) instances** of the problem class in reasonable time
Physicist, Quantum Computing Researcher	… compute reasonably **good solutions to these (selected) particular instances** of the problem class in very short time

However, the above is at least unprecise, because NP-hardness refers to decision problems.
Therefore, we should replace proven optimal solutions by **proven correct answers** and
good solutions by … **likely correct answers most of the time?**

What are Stochastic Local Search Heuristics

Stochastic	= we do something random
Local	= we do it near to where we are
Search	= we look around
Heuristics	= we do not expect a perfect outcome

What does “local” mean?

Example: **Evolutionary Algorithm**

Wanted: A fierce mamal

Wait 10 million years and you might get ...

The **WOMBAT**

Size: up to 120 cm Weight: up to 40kg

Sharp crescent-shaped claws, sharp teeth.

When attacked, can summon immense reserves of strength; one defense of a wombat against a predator underground is to crush it against the roof of the tunnel, thus suffocating the animal. Its primary defense is its toughened rear side.

Globale Optimum

Galactic Optimum

Bio Inspired

With this great power and mystery, tribal cultures worshipped tigers, bestowing them with powers that extend far beyond those of any worldly creature. For millennia, medicine men have ascribed magical powers and medicinal properties to them, and somehow, this cat became a universal apothecary. It was believed that by ingesting it, you absorb an animal's life force, its vigor, strength, and attributes.
https://blog.nationalgeographic.org/2014/04/29/tigers-in-traditional-chinese-medicine-a-universal-apothecary/

Stochastic local search heuristics

- Evolutionary algorithms
- Genetic algorithms
- Simulated annealing
- Ant-colony optimization
- Intelligent water drops algorithm (IWD)
  which mimics the behavior of natural water drops to solve optimization problems
- Slime molds: "*Slime mold algorithm: A new method for stochastic optimization*"
- Naked mole-rats: "*The naked mole-rat algorithm*"
- …

"Inspired" means, it helped to find a catchy name, that leads to positive (but wrong) associations.

These types of algorithms mostly converge to **local optima**.

Two basic optimization (search) algorithms

Space-Explorer (Discrete)

- ▷ Enumerate the feasible points in the solution space either explicitly or implicitly
- ▷ Typically, by some kind of search tree

Heuristic Idea: Just hop around randomly in the feasible region

Important: How to implicitly exclude as much of the space as possible.

Nice: Convex hull of the feasible points is known.

Path-Finder (Continuous)

1. Select a starting point a_0
2. While stopping criteria not fulfilled do
3. Find a direction d
4. Find a step length α
5. Move to new point $a_{i+1} = a_i + \alpha \cdot d$

Important: How to find a good starting point, how to compute good direction and step length?

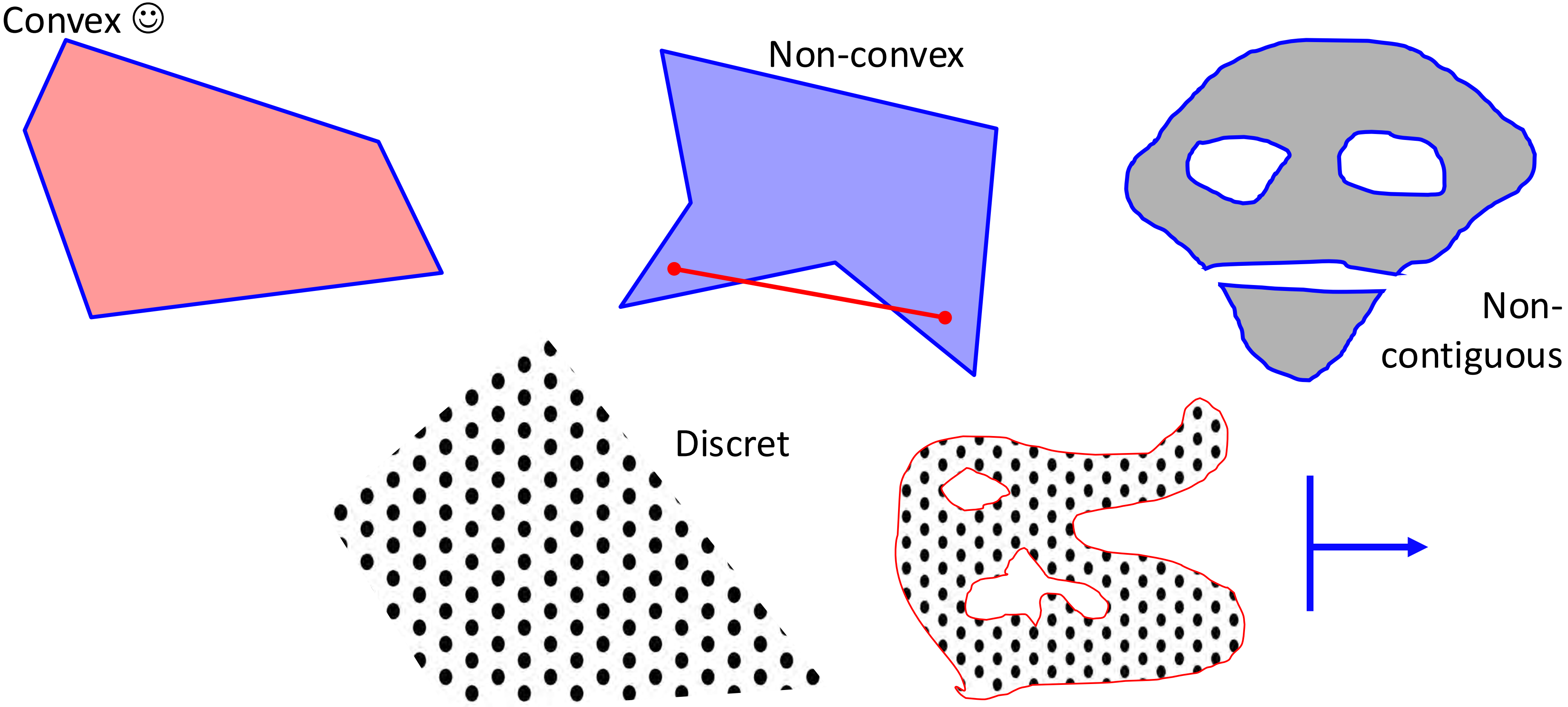
Nice: Convex region and gradient available.

What do we know about our feasible region?

How can we evaluate the objective?

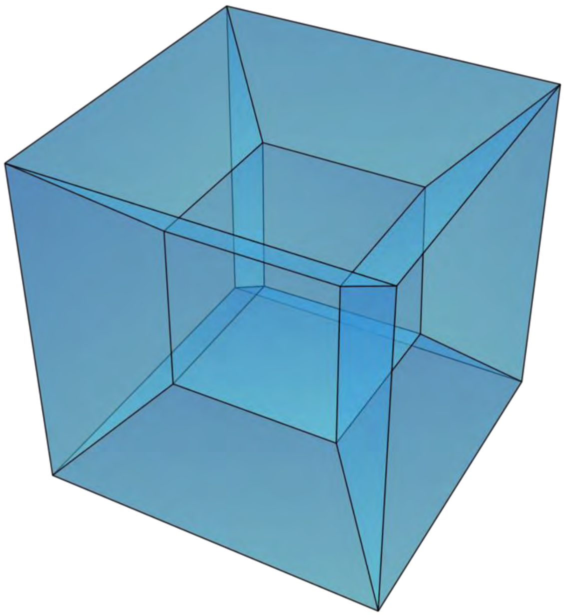
Will we get a local or a global optimum?

Shapes of feasible sets



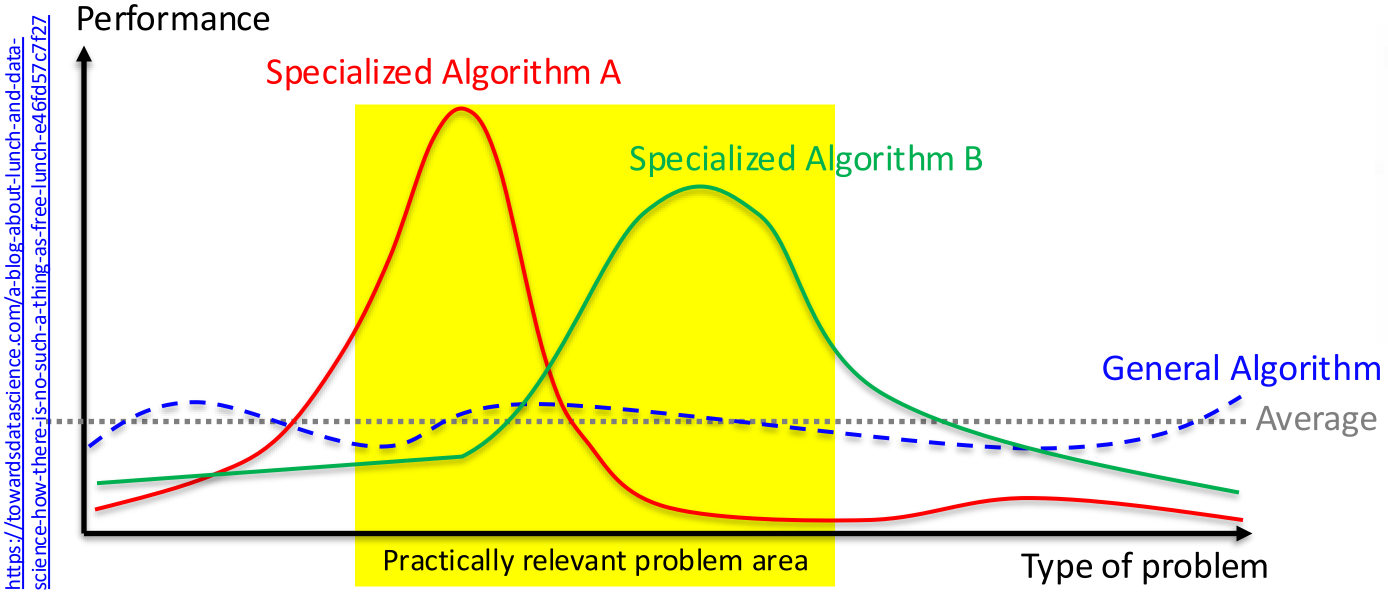
How does the convex hull look like in case of a Binary Integer Program?

Given a Binary Integer Program (BIP): $\min_{\substack{x_i \in \{0,1\}, \\ i \in \{1,\dots,N\}}} c^T x \,, Ax \leq b,$



- ▷ The solution space is an N-dimension unit-cube with 2^Nvertices.
- ▷ There are no inside integer points, therefore there can be no holes within the feasible set.
- ▷ Every feasible point is a corner of the unit-cube.
- ▷ The constraints define hyperplanes that cut of (rectangular) corners from the cube and introduce new facets.
- ▷ If we neglect dominated constraints, each constraint increases the number of vertices of the polytope.
- ▷ Fixing a variable, reduces the cube in that dimension to a point, i.e., branching completely decides a variable.

TANSTAAFL



We can trade generality for performance. This makes ensemble approaches attractive.

Wolpert, Macready: *No free lunch theorems for optimization,* IEEE T.o. Evolutionary Computation (1997), doi: 10.1109/4235.585893.

States that for a completely unstructured search space all algorithms perform equally.

Whatever one algorithm improves on one class of problems is offset by a degradation in another class.

Grover: *A fast quantum mechanical algorithm for database search*

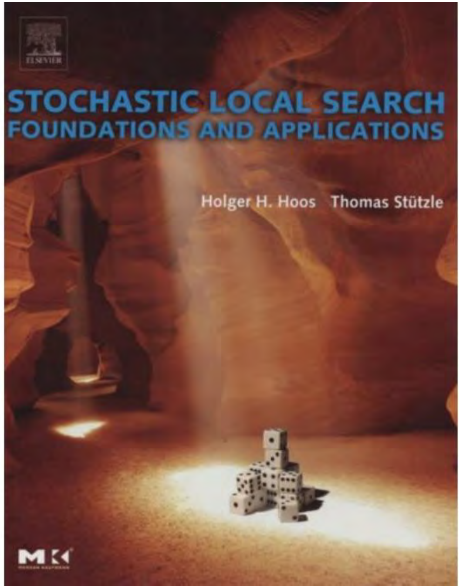
28th Annual ACM Symposium on the Theory of Computing (1996) arXiv:quant-ph/9605043v3

Grover shows a quadratic speed-up in the sense of the NFL theorem, i.e., search over an unstructured set.

If our set has any structure we can exploit, and this is usually the case for real-world optimization problems (outside crypto where the game is to hide this structure), we can achieve a practical speed-up by clever algorithms.

Stochastic local search

1. Start with some arbitrary solution
2. Repeat until time runs out:
3. Randomly change something
4. If the result is better than before, remember as current solution
5. If no improvement for some time,
   allow a change that is leading to a worse but different solution.



Notes:

- Steps 3 and 5 are obviously the key to success.
  From theoretical side important whether you can reach your entire space by the changes.
- Works best with problems where feasibility is easy to achieve.
- Usually important that iterations are fast.
- Easy to implement.
- Pretty good idea, if you have not much (mathematical) understanding/knowledge of your problem.

1-Opt, 2-Opt, n-Opt

Assume a constraint optimization problem:

$$\min_{x \in \{0,1\}^n} f(x) \text{ with } x \in X$$

- 1-opt: Starting from some random vector $\dot{x}$ we flip every $\dot{x}_i, i \in \{1, \dots, n\}$ and whenever the objective improves, we keep it, otherwise we flip back. We continue until nothing changes anymore.
- 2-opt: Now we do the same for any pair of variables $\dot{x}_i, \dot{x}_j$ with $i, j \in \{1, \dots, n\}$ and $i \neq j$.
- Chaining: We start with one flip, then do a 2nd flip and as long as the solution is not getting worse than the one, we started with, we add more flips.

You can assume it is very difficult for a human to spot improvements beyond 2-opts.

1. Let $s = s_0$
2. For k=0 to k_max:
3. Set temperature: $T \leftarrow \text{temperature}(1 - \frac{k+1}{k_{max}})$
4. Pick a random neighbor: $s_{new} \leftarrow \text{neighbour}(s)$
5. $obj\_diff = \text{objective}(s) - \text{objective}(s_{new})$
6. If $obj\_diff > 0 \text{ or } e^{obj\_diff\,/\,T} > \text{random}(0,1)$
7. $s \leftarrow s_{new}$
8. Output s

Tabu Search

1. Let $s = s_0$
2. Let $T = \emptyset$
3. Let $k = 1$
4. Repeat until finished:
5. $\quad N = \{n \in \text{neighbor(s)}\} \setminus T$
6. $\quad$ Find best neighbor: $s_{new} \leftarrow \underset{n \in N}{\operatorname{argmin}} \text{objective}(n)$
7. $\quad$ Add to tabu-list: $T = T + (s_{new}, k)$
8. $\quad k = k + 1$
9. $\quad$ Update: Remove all (n, m) from T where $m < k + T_{max}$
10. Output s

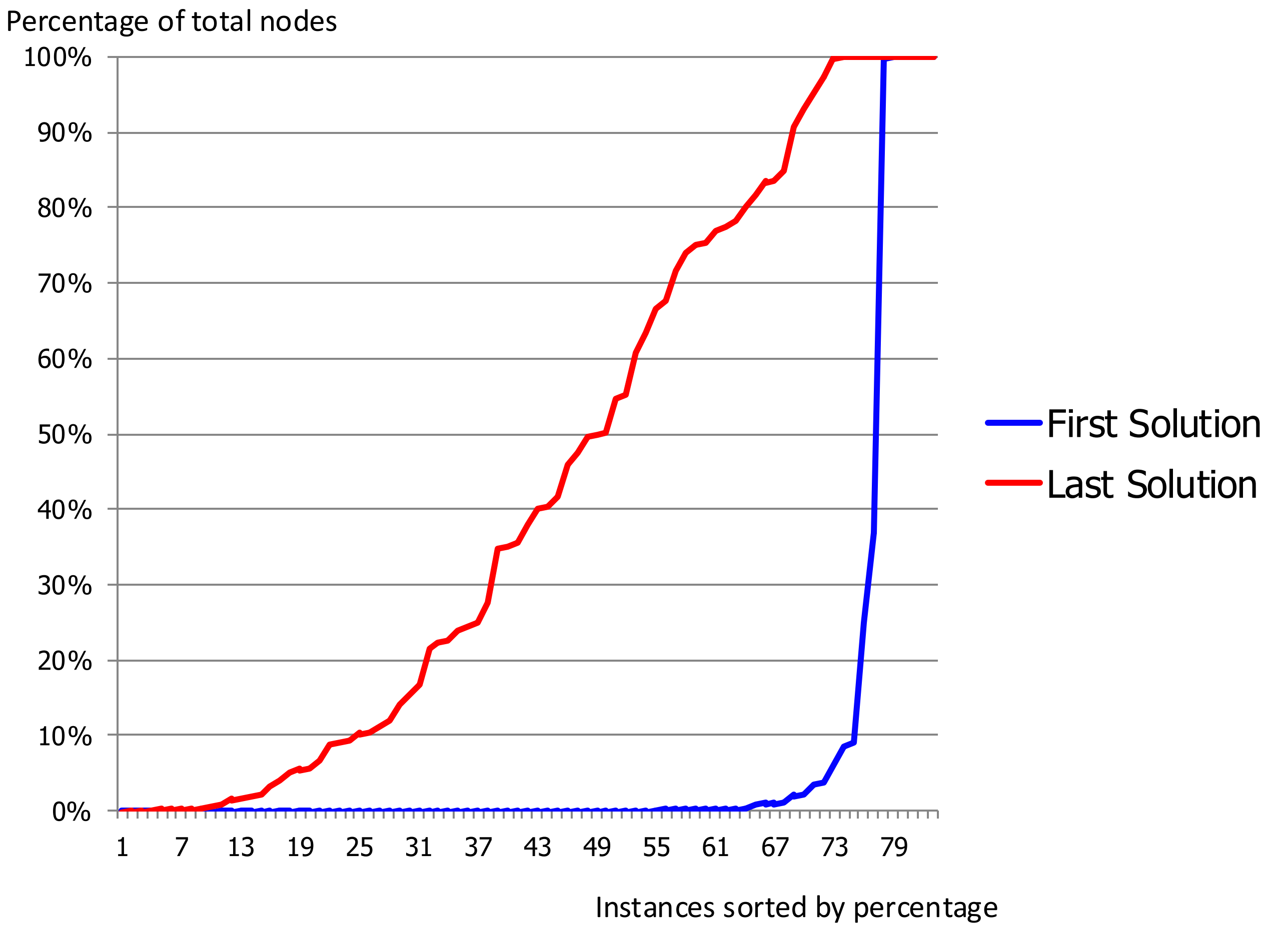
Evolutionary Algorithm

1. Generate the initial population of individuals randomly. (First generation)
2. Repeat the following regenerational steps until termination:
3. Evaluate the fitness of each individual in the population
4. Select the fittest individuals for reproduction. (Parents)
5. Breed new individuals through crossover and mutation operations to give birth to offspring.
6. Replace the least-fit individuals of the population with new individuals.

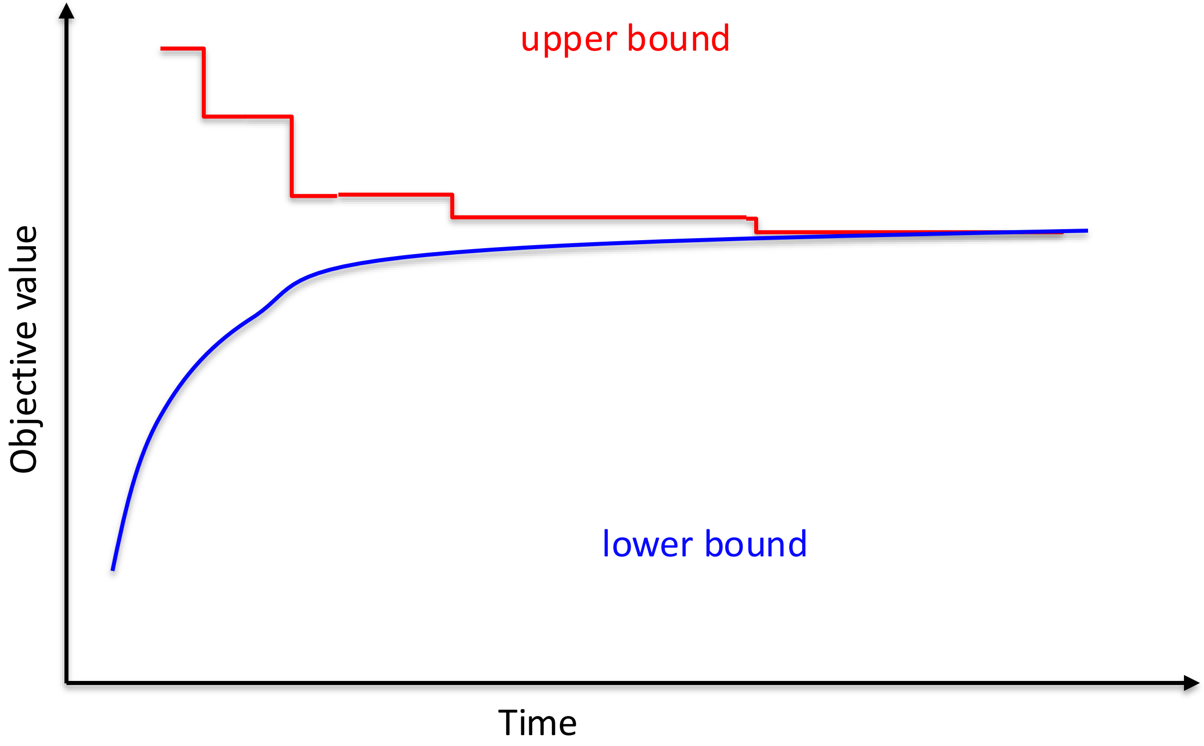
Crossover: Take (larger) parts of each ”parent” and put together.

Mutation: Randomly flip variables

Nodes to primal solution



Typical bound development



QUBO

QUBO : Quadratic Unconstraint Binary Optimization
UBQP : Unconstrained Binary Quadratic Program
(BIQ : Binary Integer Quadratic problem)

$$\min_{x\in\{0,1\}^n} x^T Q x$$

- x is a vector of binary variables, Q is a square $n \times n$ matrix of constants
- Since QUBOs are unconstraint, any 0/1 vector is a feasible solution
- All QUBOs can be brought to the form where Q is symmetric or upper triangular
- Solving QUBO (in general) is NP-hard
- Since x is binary, $x_i = x_i^2$ holds $\Longrightarrow$ The coefficients of the linear terms of the objective function correspond to the diagonal entries of Q

BIP

$$\min_{x\in\{0,1\}^n} c^T x \quad \text{s.t. } Ax \leq b$$

QUBO

$$\min_{x\in\{0,1\}^n} c^T x^2 + P(Ax + Is - b)^2$$

BIPs can be reformulated as QUBOs by putting the constraints into the objective with a penalty term P. The penalty should be zero if and only if the constraint is fulfilled.

Glover, Kochenberger, Du (2019):
A Tutorial on Formulating and Using QUBO Models
arXiv:1811.11538

Max-Cut

Graph formulation $G = (V, E, w)$

$$\max_{S,T} \sum_{i \in S, j \in T} w_{ij} \quad \text{with } S \subset V, T \subset V, S \cap T = \emptyset, S \cup T = V$$

Ising formulation

$$\max \frac{1}{4} \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij}(1 - x_i x_j), x_k \in \{1, -1\} \text{ for } k \in \{1, \dots, n\}$$

Binary Linear Programing formulation:

$$\max \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} z_{ij}$$
$$z_{ij} \leq x_i + x_j$$
$$z_{ij} \leq 2 - (x_i + x_j)$$
$$x_k \in \{0,1\}$$
$$z_{ij} \in \{0,1\}$$

Binary Quadratic Programing formulation:

$$\max \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij}(x_i + x_j - 2x_i x_j)$$
$$x_i \in \{0,1\}$$

Can be written as:

$$\min_{x \in \{0,1\}^n} x^T Q x$$

Hans Mittelmann: Nonconvex QUBO-QPLIB Benchmark, 10.07.2023

prob#	Baron	SCIP	McSparse	Octeract	Gurobi	QuBowl	BiqBin	SHOT	ABS2
3506	87		407	89	165	102		789	0.047
3565	2	52	13	2	5	1	12	3	0.028
3642							f		0.063
3650									0.219
3693							f		*0.899*
3705	5	94	20	7	6	1	232	6	0.043
3706	881		2418	899	2357	984			0.096
3738	58	2753	137	73	94	29		197	0.096
3745	24	442	55	29	14	12		24	0.043
3822									0.051
3832	218		465	244	260	245			0.252
3838									7.971
3850							f		1.844
3852	1	35	46	2	2	2	411	2	0.020
3877	164		448	156	322	36			0.044
5721					1	1	2026	1	*7.509*
5725	10	81	4	10	7	1	39	1	0.122
5755	2	3	1	1	1	1	1317	1	1.061
5875						1720	1150		0.002
5881	440	2231	75	120	47	29	10	64	0.001
5882						607	232		0.000
5909							1842		0.019
5922									0.022
mean	1.85	5.65	2.44	1.80	1.46	1.00	5.31	2.06	
solved	12	8	12	12	13	15	10	11	

QPLIB A Library of Quadratic Programming Instances

https://qplib.zib.de

http://plato.asu.edu/ftp/qubo.html

The codes were run on an AMD Ryzen 9 5900X (12 cores, 128GB) on the 23 unconstrained binary problems from QPLIB. All problems were solved GLOBALLY. Times given are elapsed times in seconds, time limit 1hr; 12 threads. Shifted and scaled geometric mean of runtimes:.

Baron-23.6.22	https://www.minlp.com/baron
SCIP-8.0/cplex	http://scipopt.org
OCTERACT-4.7.1/cplex	https://octeract.com
Gurobi-10.0.1	http://gurobi.com
QuBowl	https://arxiv.org/pdf/2202.02305.pdf
Biqcrunch-2	https://biqcrunch.lipn.univ-paris13.fr
BiqBin	http://www.biqbin.eu
McSparse-2.0	http://mcsparse.uni-bonn.de
SHOT-1.1/gurobi	https://shotsolver.dev/shot
ABS/Amplify	https://amplify.fixstars.com/en

ABS2 on RTX4090 (red = suboptimal solution, gap 0.17, 0.09)

Fragen

有問題嗎

คำถาม

?

質問

Вопросы

Questions

Câu hỏi

Thank you very much!