Artificial Intelligence

Local Search

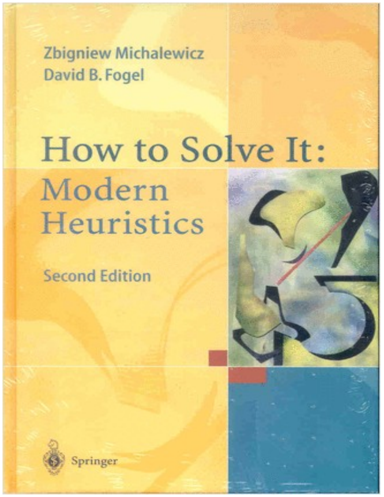
Fred Koriche

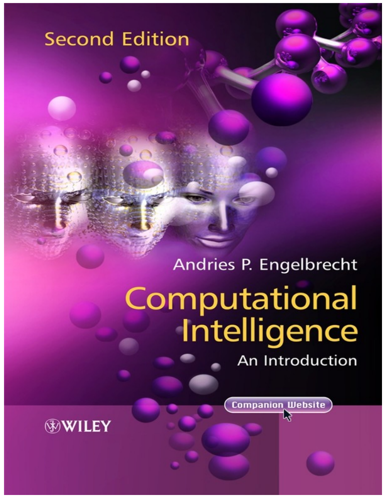
koriche@lirmm.fr

Overview

I. Optimization Problems
II. Local Search
III. Hill Climbing
IV. Simulated Annealing
V. Evolutionary Algorithms

Books





Optimization Problems

Problem

Search Space

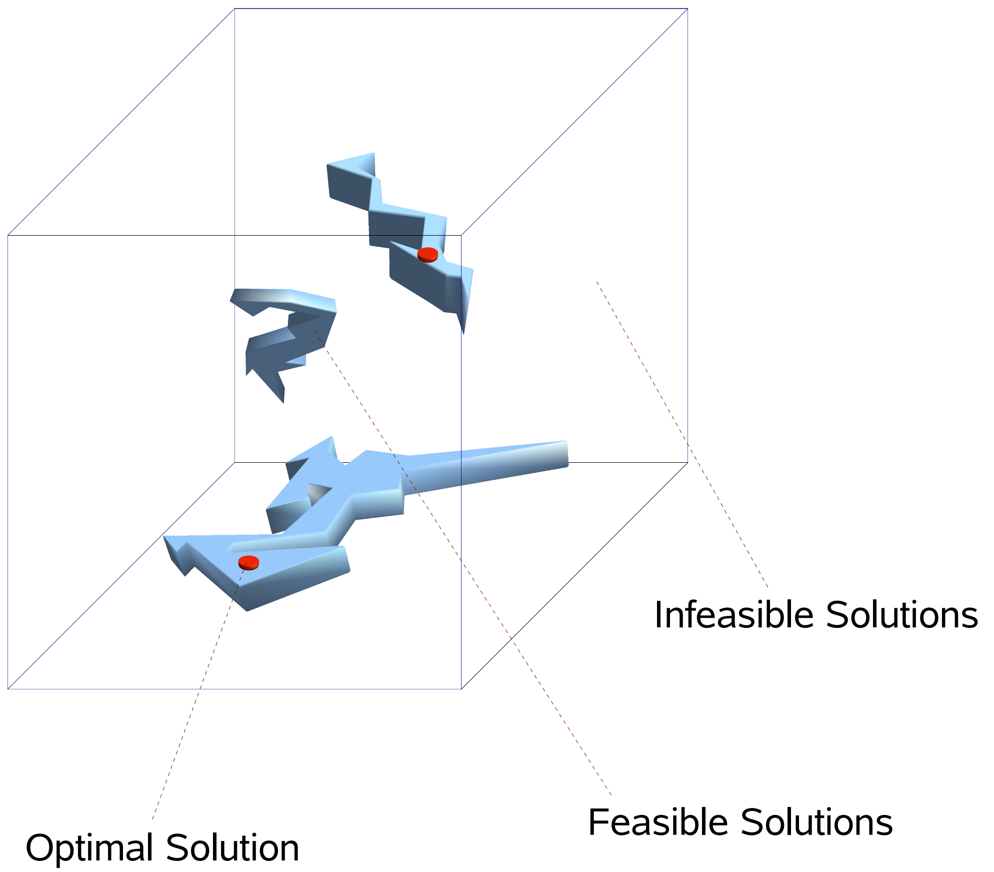
Space S of all possible solutions

Constraints

Separate feasible from infeasible solutions

Objective

Separate optimal from feasible solutions



Optimization Problems

N-Queens

Solution

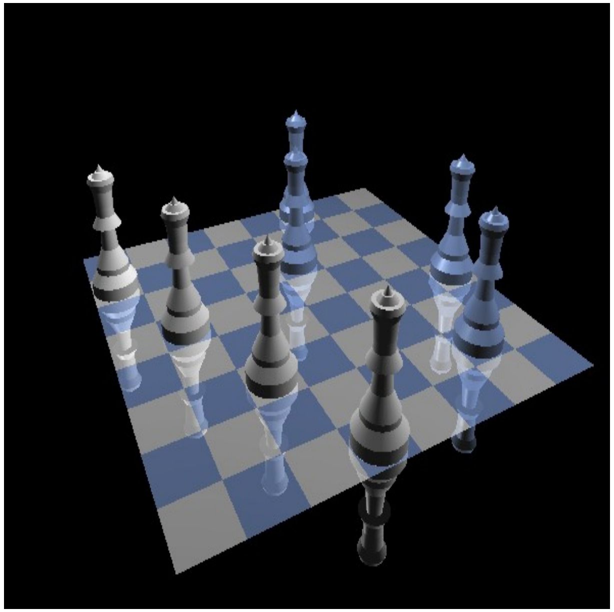
Any placement of the N queens on the chess board

Constraints

Two queens cannot be on the same column

Objective

Minimize the number of queens attacking each other



Optimization Problems

Pathfinding

Solution

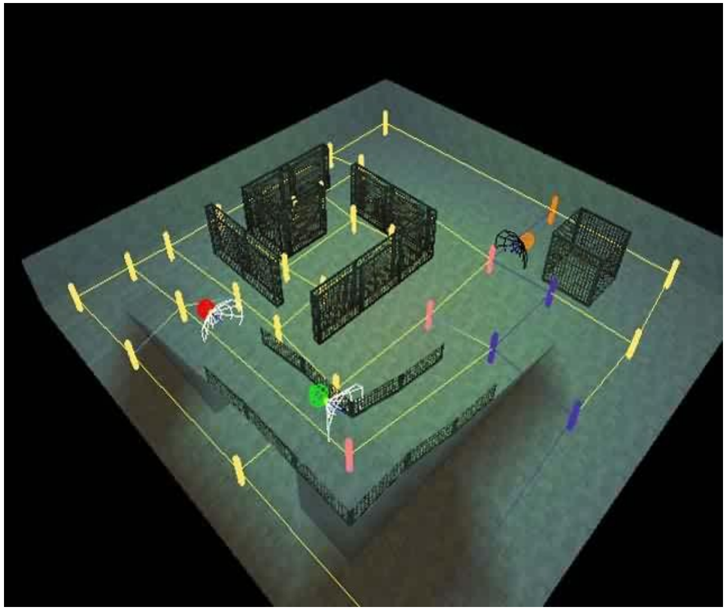
Any path in the graph of waypoints starting from the initial position

Constraints

The path must reach a goal node

Objective

Minimize the cost function of each state



Optimization Problems

Constraint Optimization

Solution

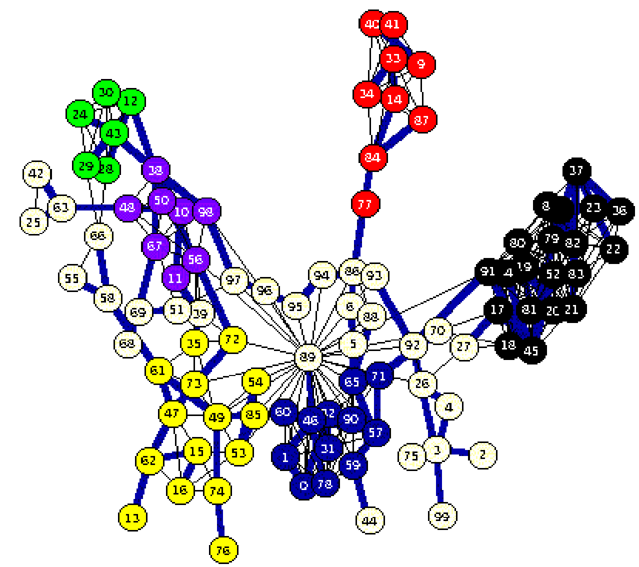
Any assignment of variables to discrete values

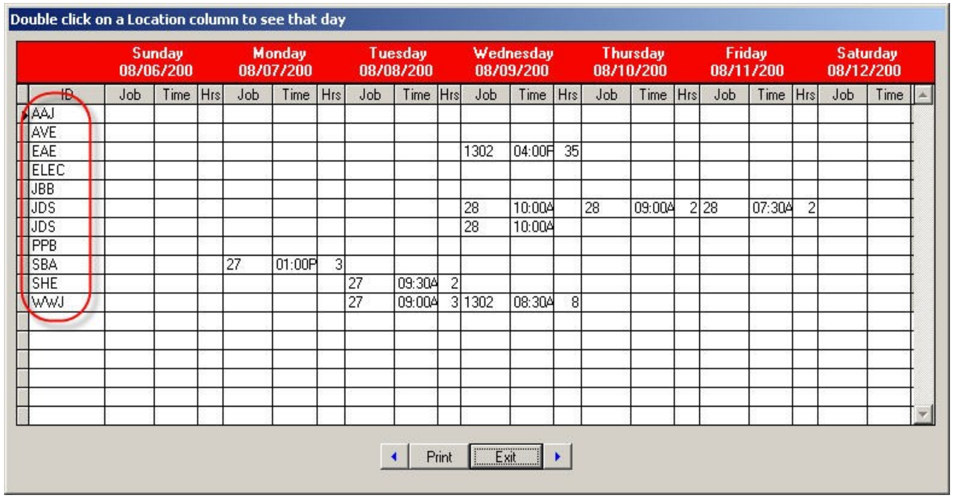
Constraints

A set of **rules** that must be satisfied

Objective

Maximize the set of **preferences**





Double click on a Location column to see that day

ID	Sunday 08/06/200			Monday 08/07/200			Tuesday 08/08/200			Wednesday 08/09/200			Thursday 08/10/200			Friday 08/11/200			Saturday 08/12/200	
	Job	Time	Hrs	Job	Time	Hrs	Job	Time	Hrs	Job	Time	Hrs	Job	Time	Hrs	Job	Time	Hrs	Job	Time
AAJ																				
AVE																				
EAE										1302	04:00F	35								
ELEC																				
JBB																				
JDS										28	10:00A		28	09:00A	2	28	07:30A	2		
JDS										28	10:00A									
PPB																				
SBA				27	01:00P	3														
SHE							27	09:30A	2											
WWJ							27	09:00A	3	1302	08:30A	8								

Print Exit

Optimization Problems

Non Linear Programming

Solution

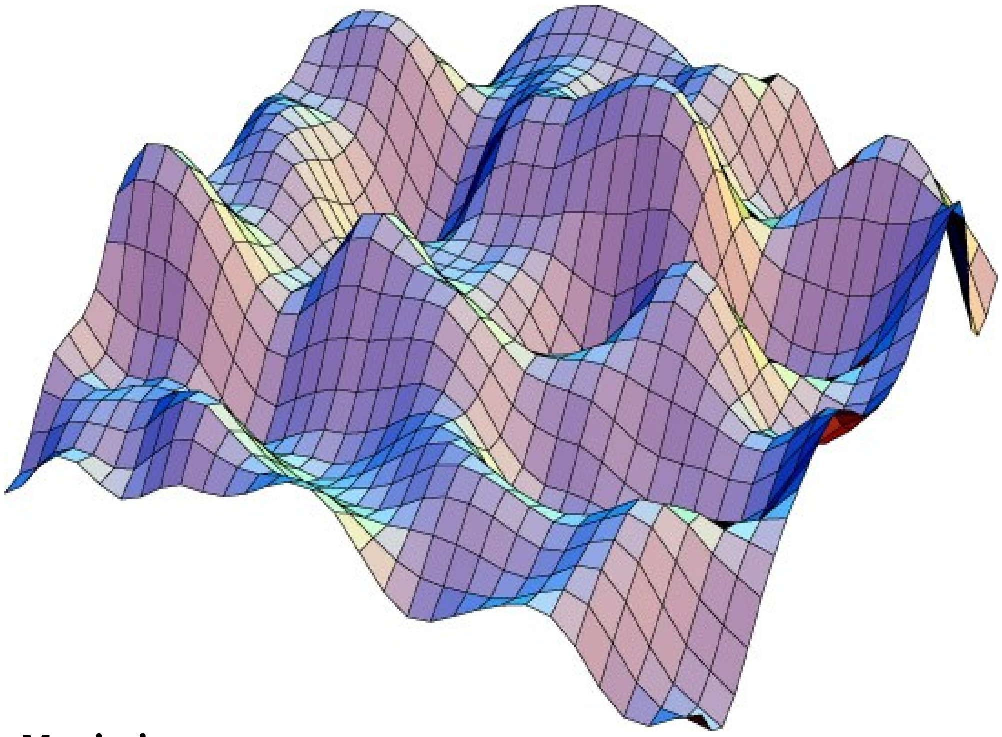
Any assignment of variables to real values

Constraints

A set of **rules** that must be satisfied

Objective

Minimize (or maximize) a **function** on the variables



Maximize

$$\left|\frac{\sum_{i=1}^{n}\cos^4(x_i)-2\prod_{i=1}^{n}\cos^2(x_i)}{\sqrt{\sum_{i=1}^{n} i x_i^2}}\right|$$

Subject to

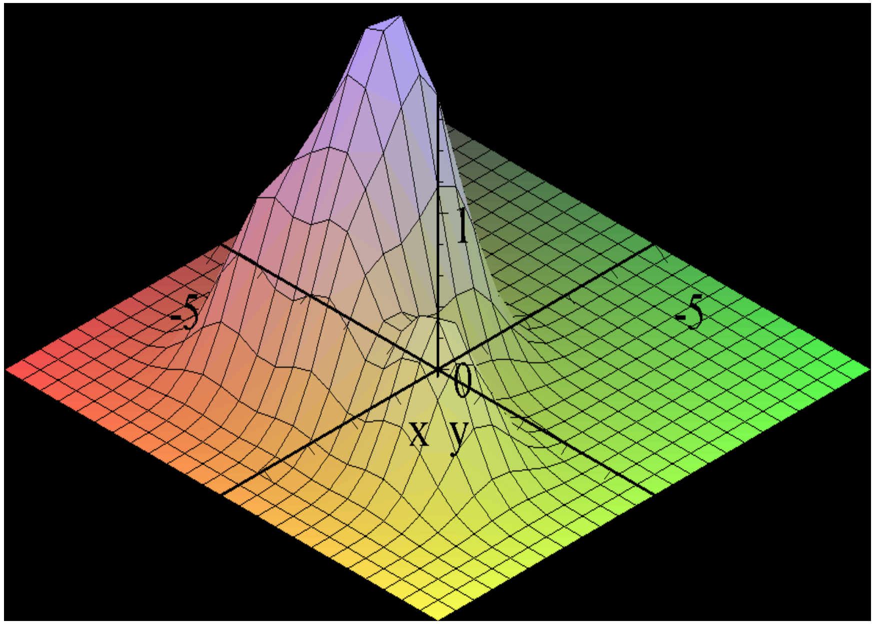
$$\prod_{i=1}^{n} x_i \geq 075,\ \sum_{i=1}^{n} \leq 7.5,\ 0 \leq x_i \leq 10$$

Local Search

Local Search

1. Pick a solution and evaluate it.
2. Apply a local transformation to generate a new solution and evaluate it
3. If the new solution is better, then exchange it with the current solution.

Repeat 1-3 until no transformation improves the current solution



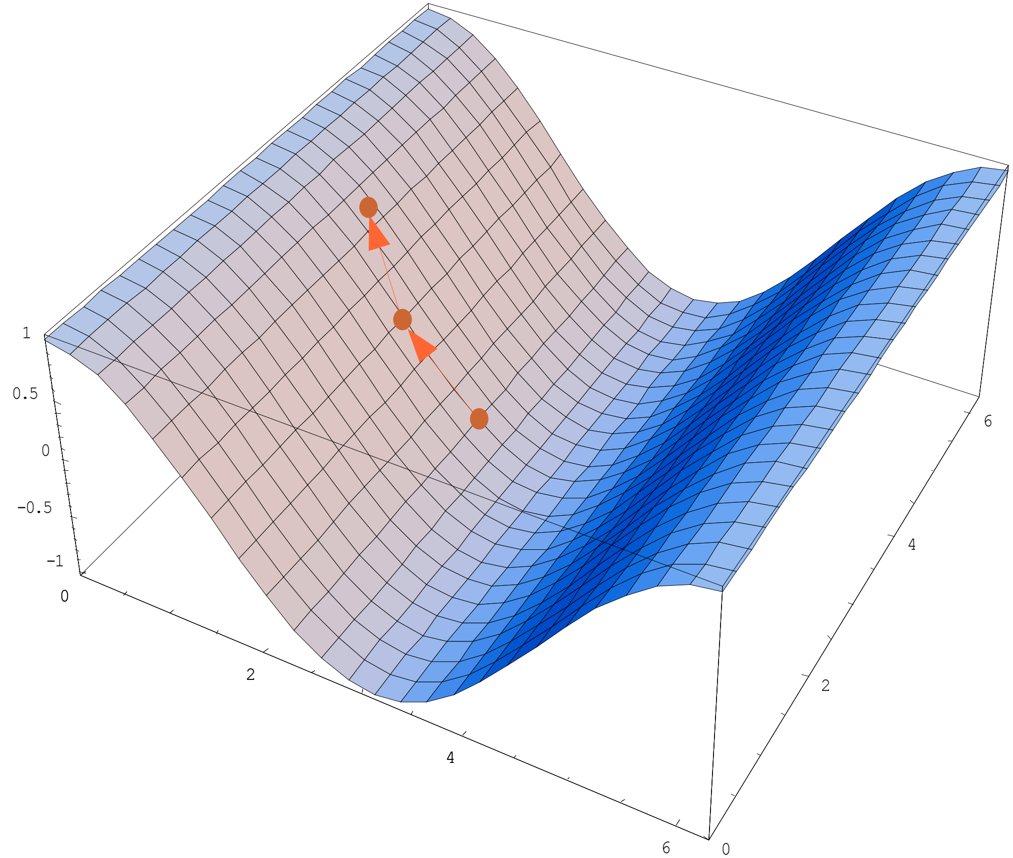
Hill-Climbing

Hill Climbing

select a point $\boldsymbol{x}$ at random;
v_x = Eval($\boldsymbol{x}$);

moves = 1;
repeat
for each point $\boldsymbol{y}$ in Neighbors($\boldsymbol{x}$)
    v_y = Eval($\boldsymbol{y}$);
    if $v_y > v_x$ **then**
        $\boldsymbol{x} = \boldsymbol{y}$;
        $v_x = v_y$;
until moves = MaxMoves;

return $\boldsymbol{x}$;

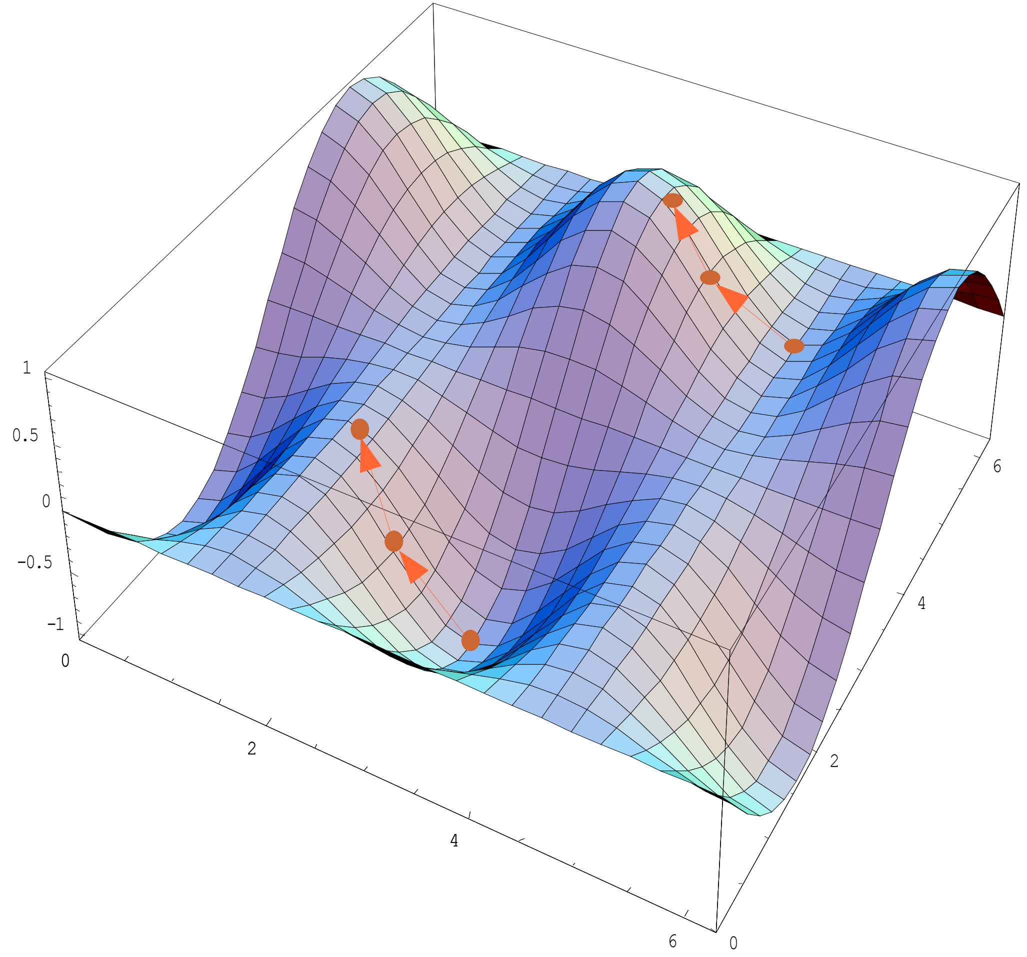


Hill-Climbing

Iterative Hill Climbing

select a point $\boldsymbol{x}$ at random;
v_x = Eval($\boldsymbol{x}$);

tries = 1;
repeat
    moves = 1;
    repeat
    for each point $\boldsymbol{y}$ in Neighbors($\boldsymbol{x}$)
        v_y = Eval($\boldsymbol{y}$);
        if $v_y > v_x$ **then**
            $\boldsymbol{x} = \boldsymbol{y}$;
            $v_x = v_y$;
    until moves = MaxMoves;
until tries = MaxTries
return $\boldsymbol{x}$;



Hill-Climbing

Local Pathfinding

Solution

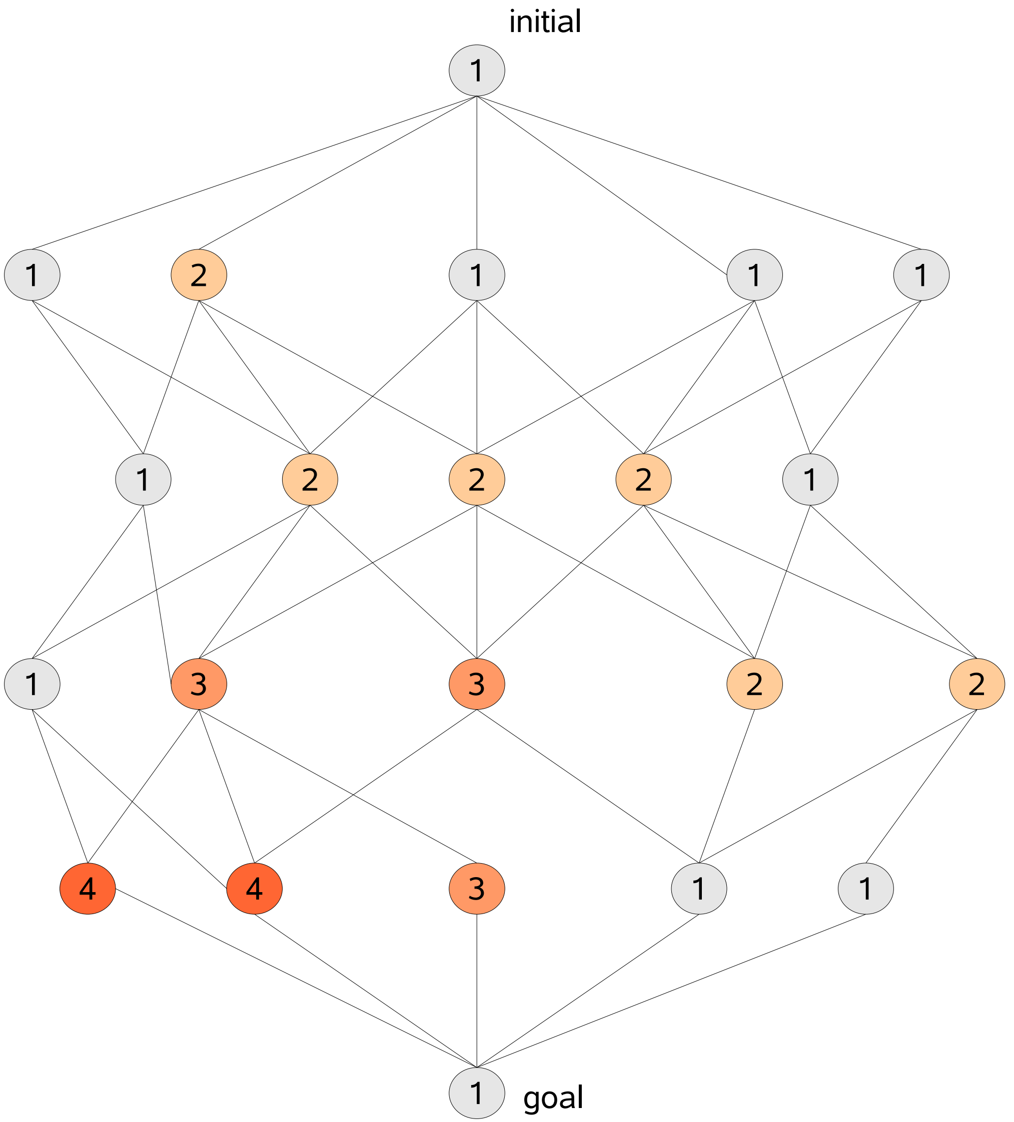
Any path from initial state to goal state

Evaluation

Cost of the path

Neighbors

Any path obtained by swapping to a new node and applying greedy search from this node to the goal



Hill-Climbing

Local Pathfinding

Solution

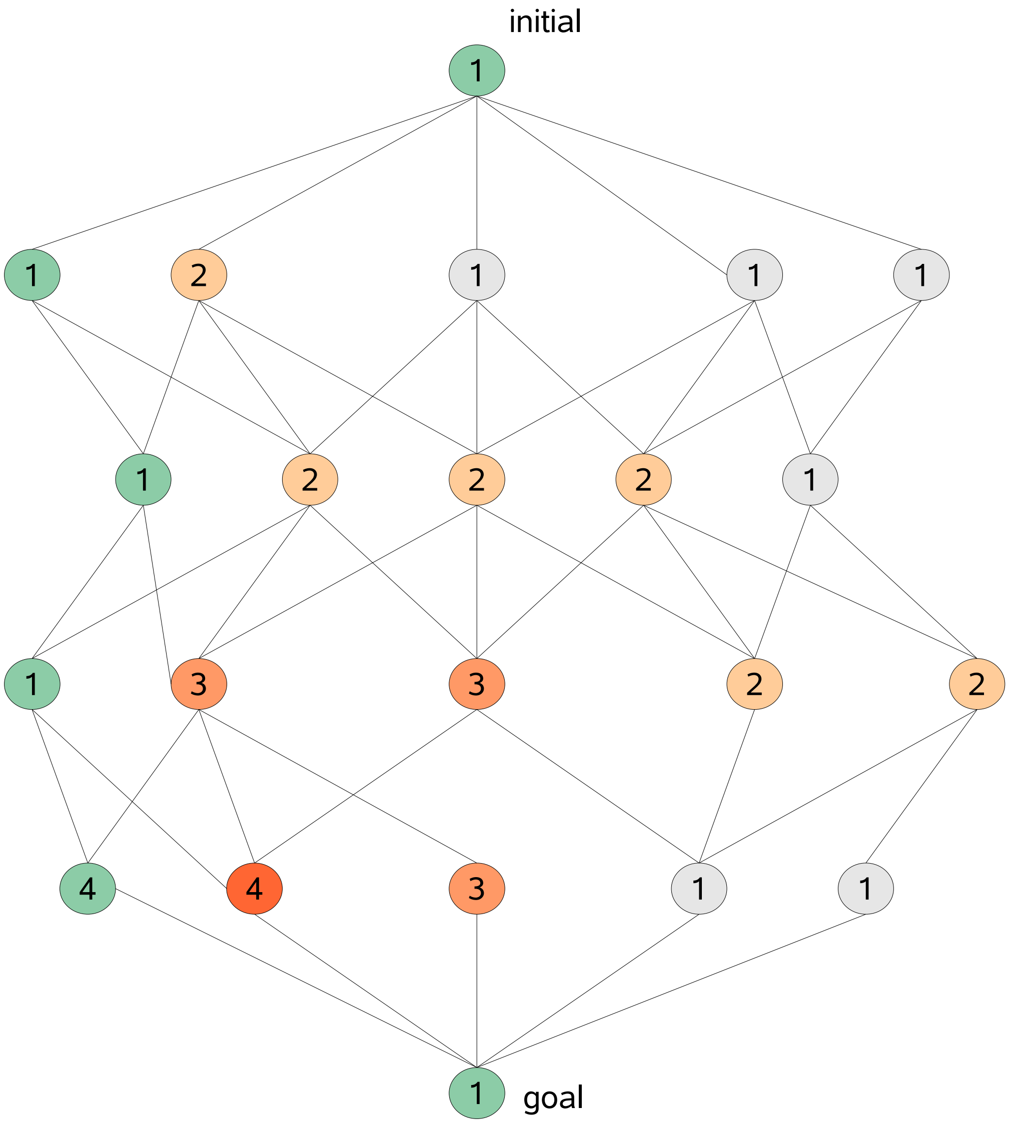
Any path from initial state to goal state

Evaluation

Cost of the path

Neighbors

Any path obtained by swapping to a new node and applying greedy search from this node to the goal



Hill-Climbing

Local Pathfinding

Solution

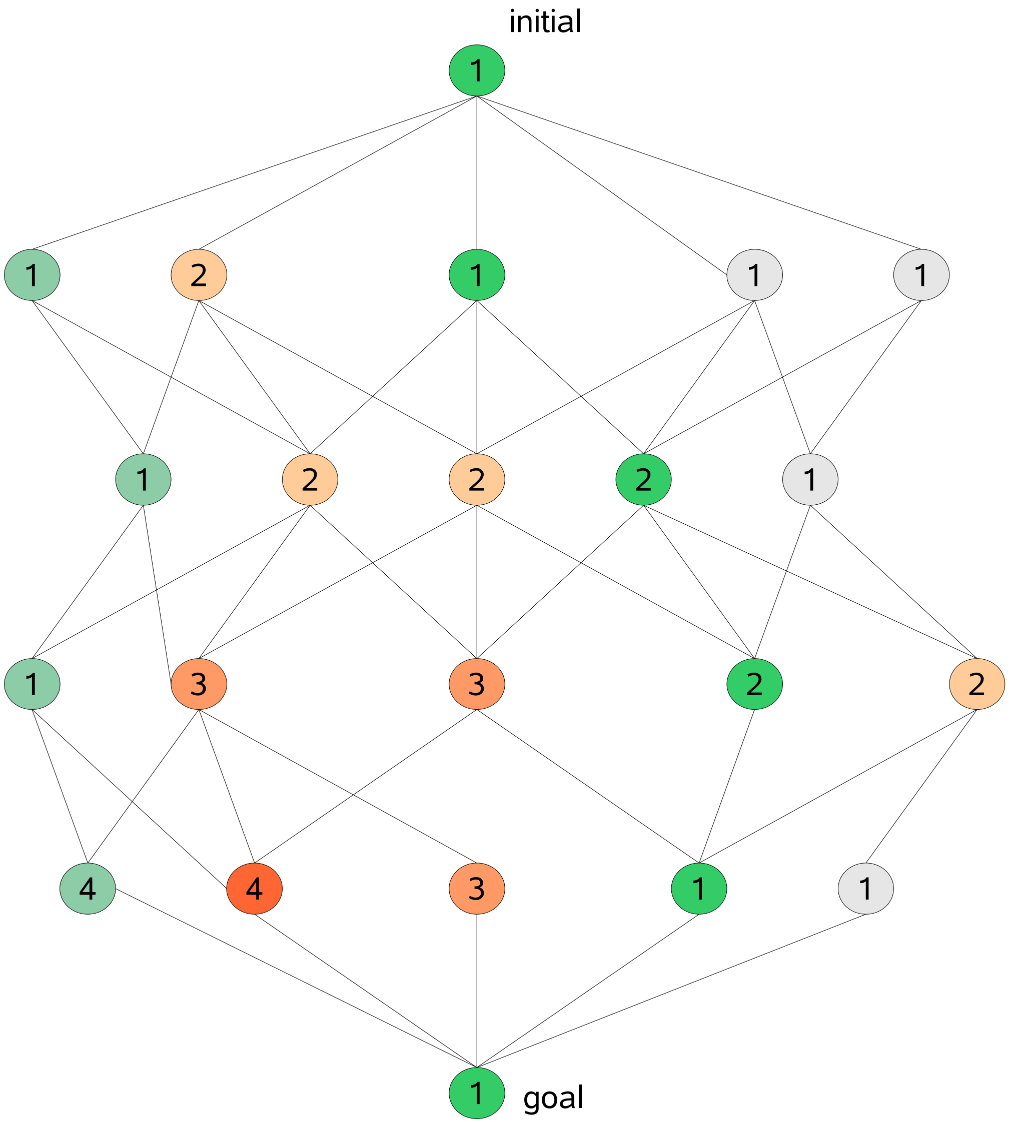
Any path from initial state to goal state

Evaluation

Cost of the path

Neighbors

Any path obtained by swapping to a new node and applying greedy search from this node to the goal



Hill-Climbing

Local Pathfinding

Solution

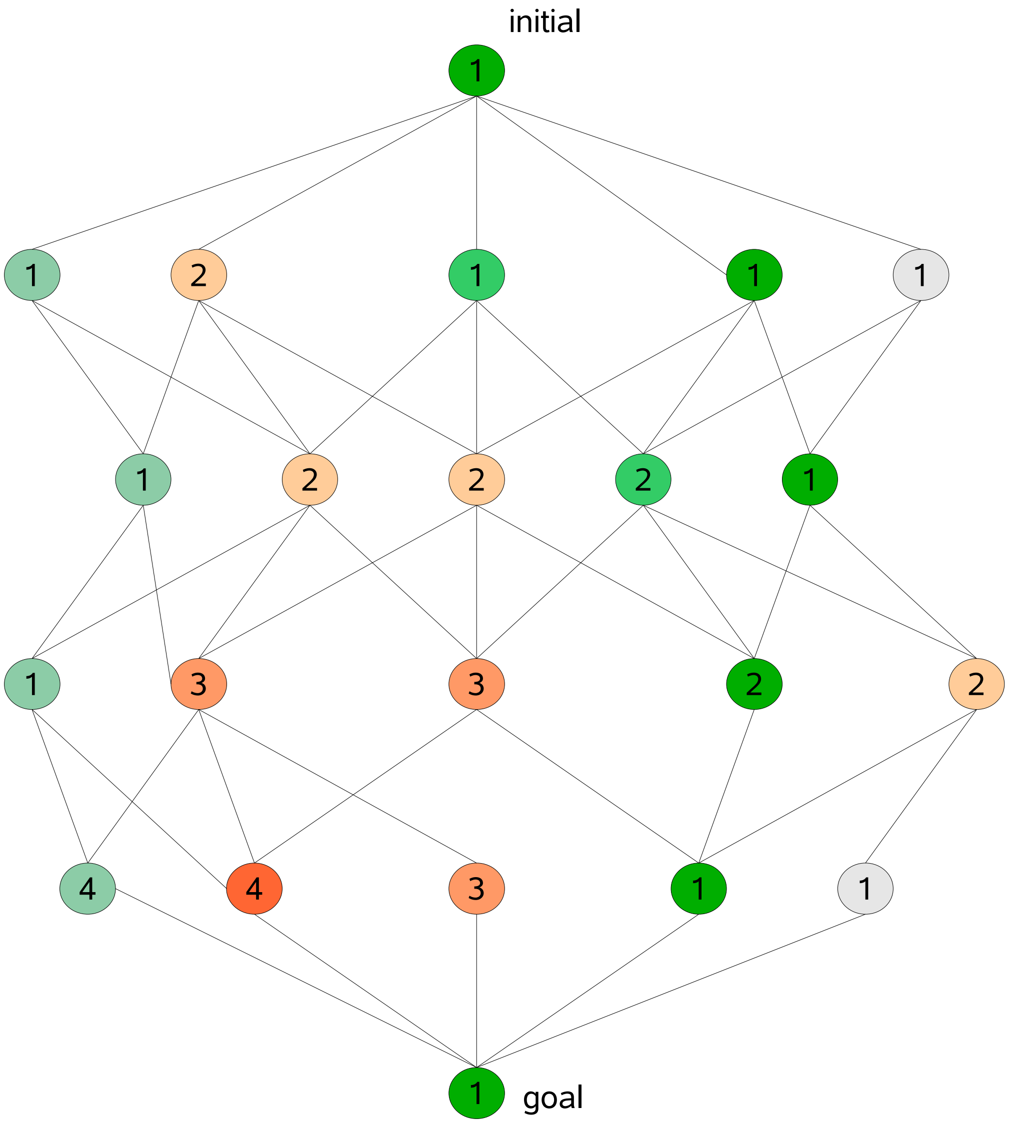
Any path from initial state to goal state

Evaluation

Cost of the path

Neighbors

Any path obtained by swapping to a new node and applying greedy search from this node to the goal



Hill-Climbing

GSAT Algorithm

Solution

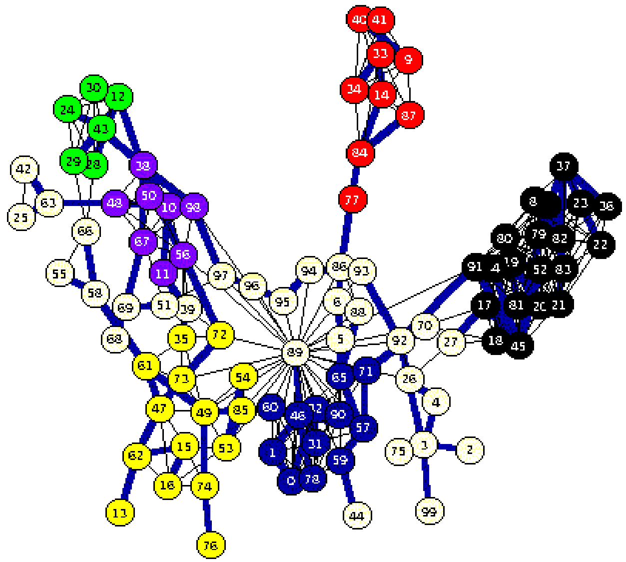
Any assignment of variables to discrete values

Evaluation

Number of clauses satisfied

Neighbors

All solutions at Hamming distance 1 from the current solution



Hill-Climbing

GSAT Algorithm

Solution

Any assignment of variables to discrete values

Evaluation

Number of clauses satisfied

Neighbors

All solutions at Hamming distance 1 from the current solution

$$(x_1 \vee \overline{x}_2) \wedge (x_2 \vee \overline{x}_3) \wedge (x_2 \vee x_4) \wedge (x_1 \vee x_3)$$

Hill-Climbing

GSAT Algorithm

Solution

Any assignment of variables to discrete values

Evaluation

Number of clauses satisfied

Neighbors

All solutions at Hamming distance 1 from the current solution

$$(x_1 \vee \overline{x}_2) \wedge (x_2 \vee \overline{x}_3) \wedge (x_2 \vee x_4) \wedge (x_1 \vee x_3)$$

	x_1	x_2	x_3	x_4
1	0	0	0	0

Hill-Climbing

GSAT Algorithm

Solution

Any assignment of variables to discrete values

Evaluation

Number of clauses satisfied

Neighbors

All solutions at Hamming distance 1 from the current solution

$$(x_1 \vee \overline{x}_2) \wedge (x_2 \vee \overline{x}_3) \wedge (x_2 \vee x_4) \wedge (x_1 \vee x_3)$$

	x_1	x_2	x_3	x_4
1	0	0	0	0

Hill-Climbing

GSAT Algorithm

Solution

Any assignment of variables to discrete values

Evaluation

Number of clauses satisfied

Neighbors

All solutions at Hamming distance 1 from the current solution

$$(x_1 \vee \overline{x}_2) \wedge (x_2 \vee \overline{x}_3) \wedge (x_2 \vee x_4) \wedge (x_1 \vee x_3)$$

	x_1	x_2	x_3	x_4
1	0	0	0	0
2	1	0	0	0

Hill-Climbing

GSAT Algorithm

Solution

Any assignment of variables to discrete values

Evaluation

Number of clauses satisfied

Neighbors

All solutions at Hamming distance 1 from the current solution

$$(x_1 \vee \overline{x}_2) \wedge (x_2 \vee \overline{x}_3) \wedge (x_2 \vee x_4) \wedge (x_1 \vee x_3)$$

	x_1	x_2	x_3	x_4
1	0	0	0	0
2	1	0	0	0
3	1	0	0	1

Hill-Climbing

Strengths

Fast algorithms

No memory

Very simple !

Weaknesses

Frequently return local optimas

No information in the deviation between the local optimum and the global optimum

Difficult to provide an upper bound on the overall computational time

Hill-Climbing

Stochastic Hill-Climbing

Simlated Annealing

Simulated Annealing

Stochastic Hill Climbing

trial = 1;

select a point $\boldsymbol{x}$ at random;
v_x = Eval($\boldsymbol{x}$);

repeat

take at random a point $\boldsymbol{y}$ in Neighbors($\boldsymbol{x}$)

v_y = Eval($\boldsymbol{y}$);

select $\boldsymbol{x} = \boldsymbol{y}$ with probability $\left(1+e^{\frac{v_x - v_y}{T}}\right)^{-1}$

until trial = MaxTrials;

return $\boldsymbol{x}$;

Idea

1. Select only one point in the neighborhood of the current solution

2. Accept this new point with some probability that depends on the relative merit of the new point

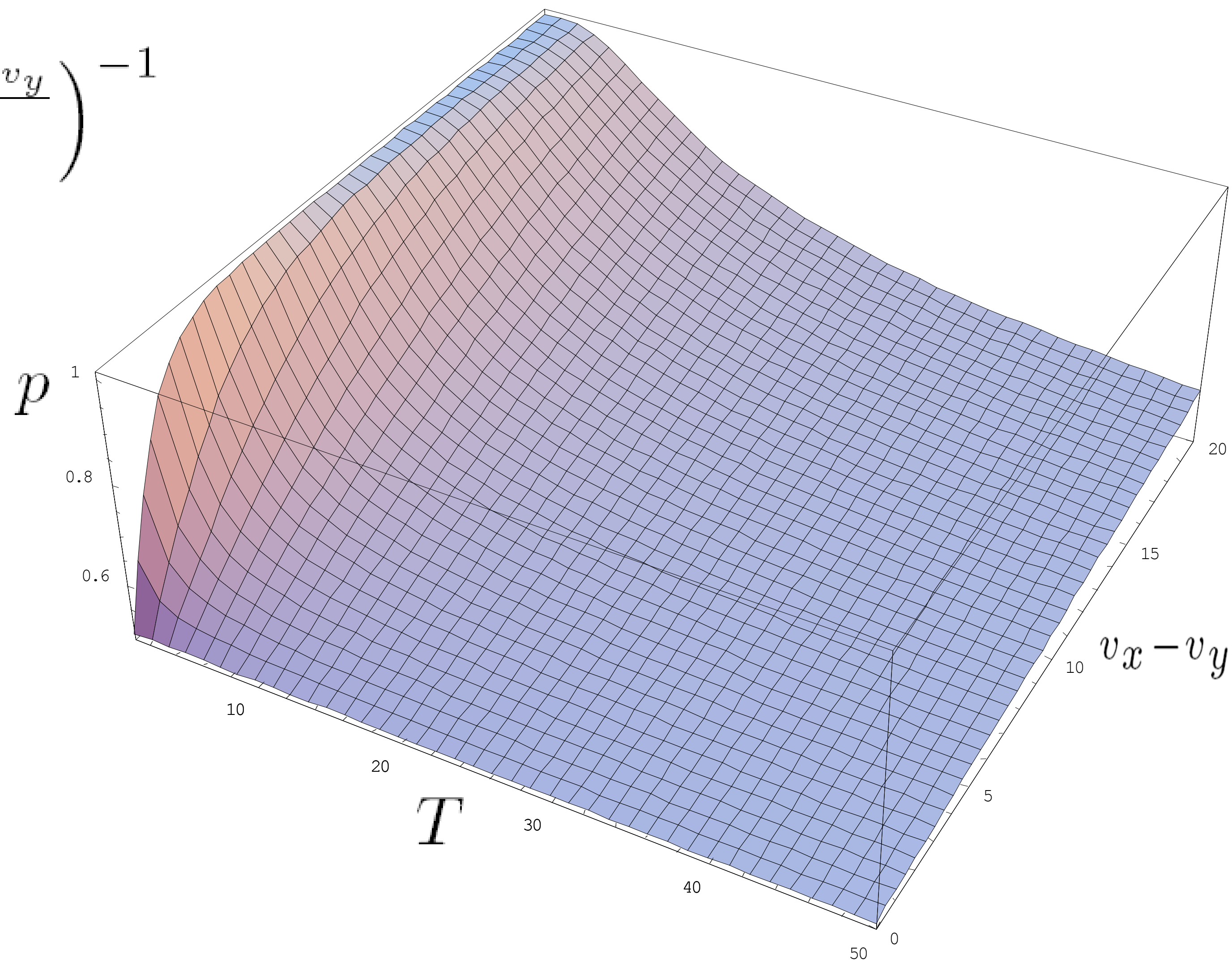
Simulated Annealing

The temperature T

$$p = \left(1 + e^{\frac{v_x - v_y}{T}}\right)^{-1}$$

Plot function with

$$v_x = 0$$
$$v_y \in [0, 20]$$
$$T \in [1, 50]$$



Simulated Annealing

Simulated Annealing

t = 1;
select a point $\boldsymbol{x}$ at random;
v_x = Eval($\boldsymbol{x}$);

repeat
  $T = T_{max}$
  repeat
    take at random a point $\boldsymbol{y}$ in Neighbors($\boldsymbol{x}$)
    if $v_x < v_y$
      then $\boldsymbol{x} = \boldsymbol{y}$;
    else
      select $\boldsymbol{x} = \boldsymbol{y}$ with probability $\left(1+e^{\frac{v_x - v_y}{T}}\right)^{-1}$
    $T = T_{max}\, e^{-tr}$;
  until $T < T_{min}$;
until t = maxTrials;

return $\boldsymbol{x}$;

Idea

1. Start with $T = T_{max}$
2. Iteratively lower T
3. If temperature is T_{min} restart with $T = T_{max}$

Simulated Annealing

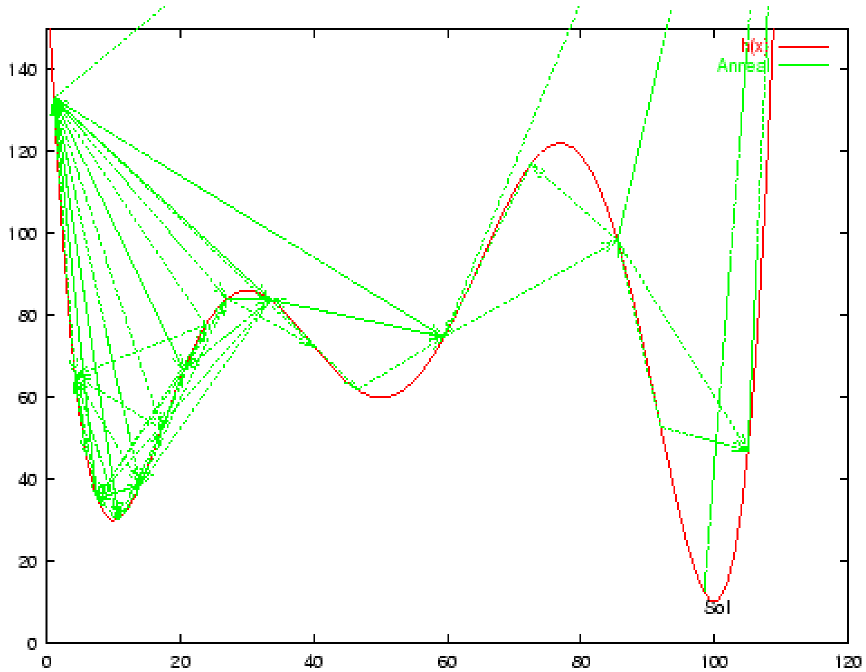
Non-Linear Programming

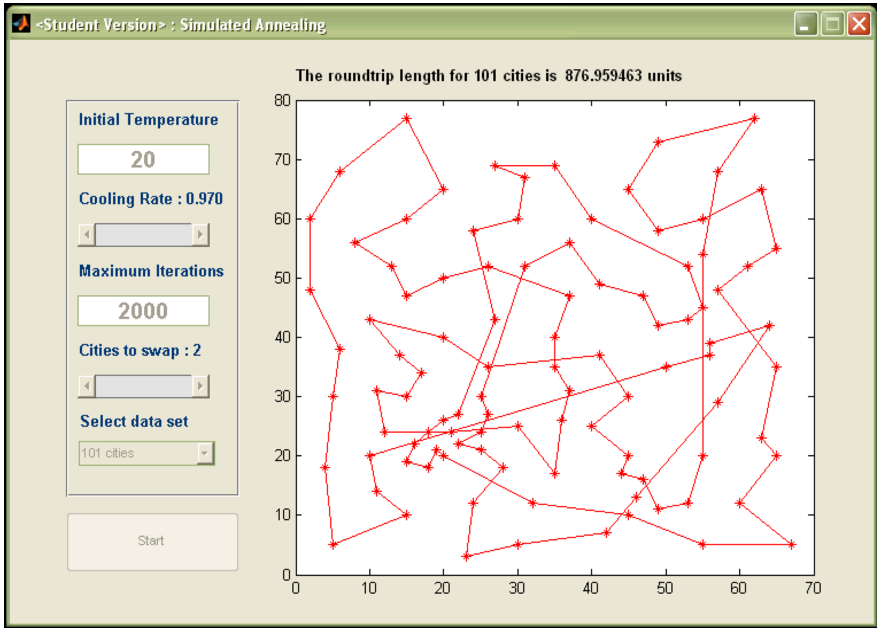
Function Minimization

Constraint Optimization

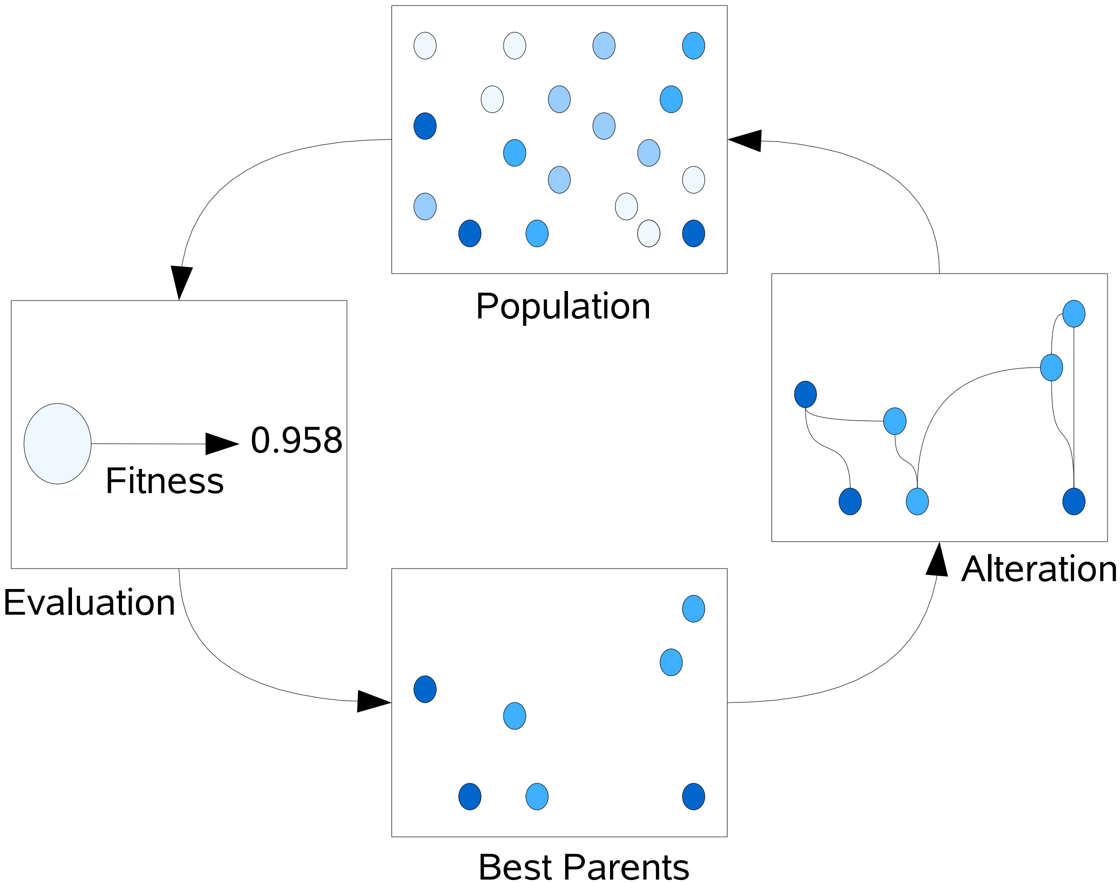
SA-SAT

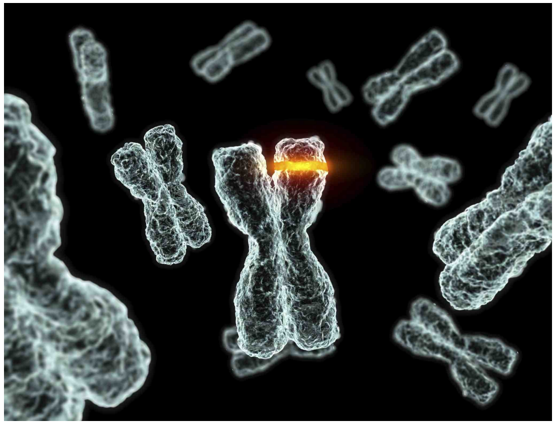
SA-TSP





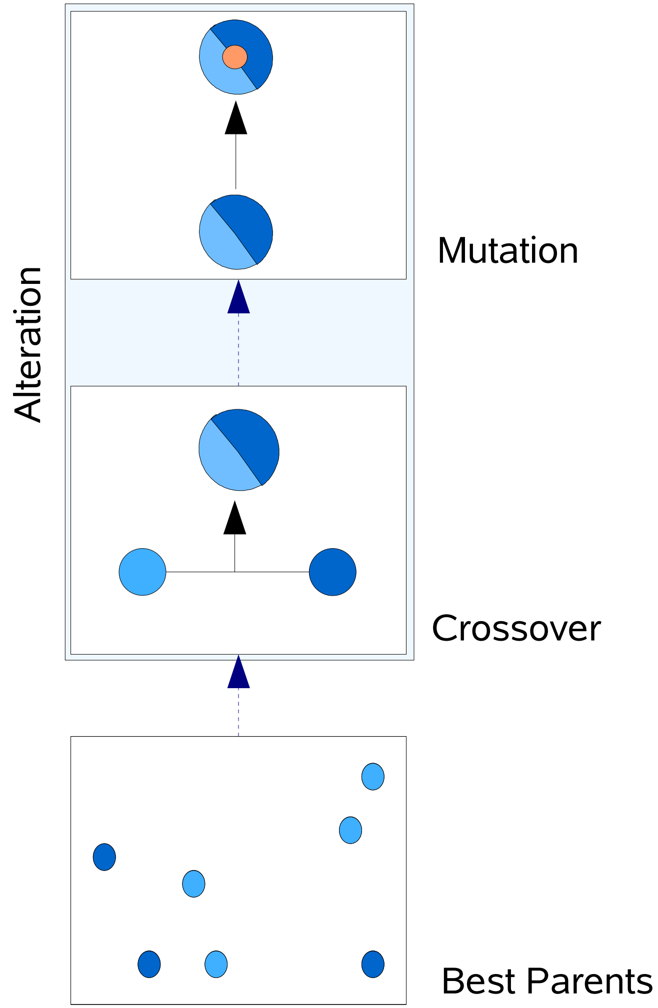
Evolutionary Algorithms

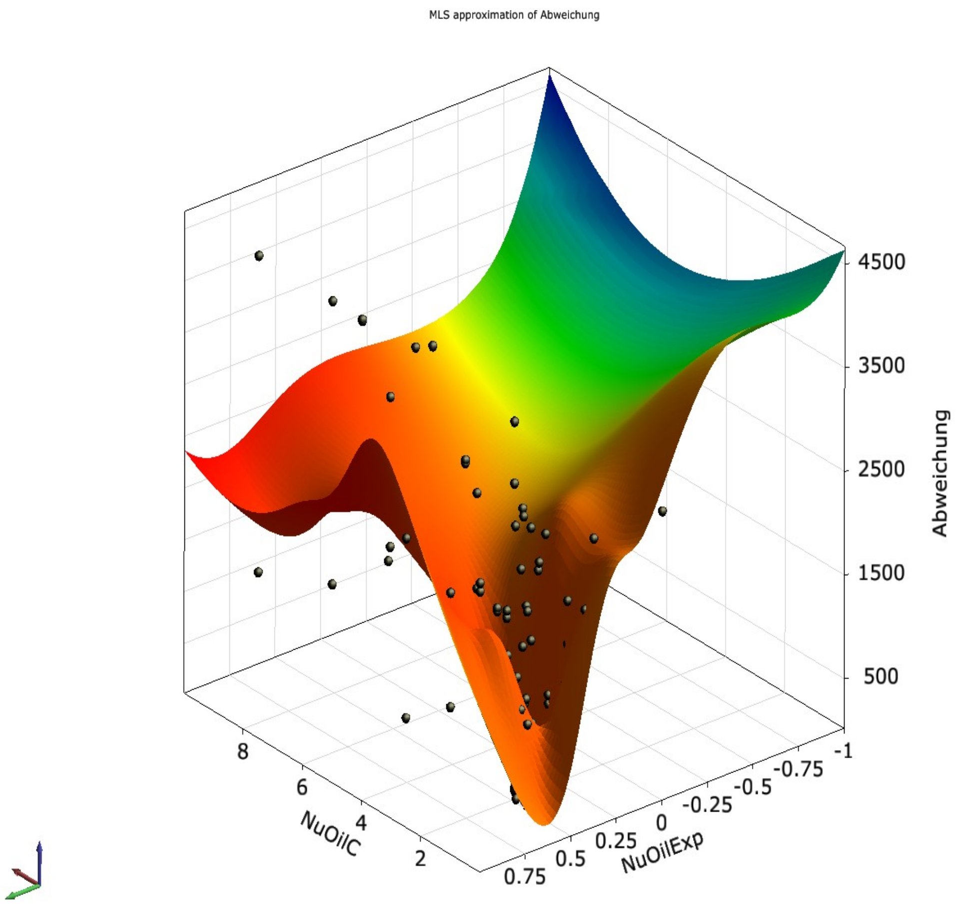




Solutions are viewed as chromosomes

Evolutionary Algorithms





Evolutionary Algorithms

Evolutionary Algorithm

t = 1;
Initialize Population P_t;

repeat

Evaluate P_t;

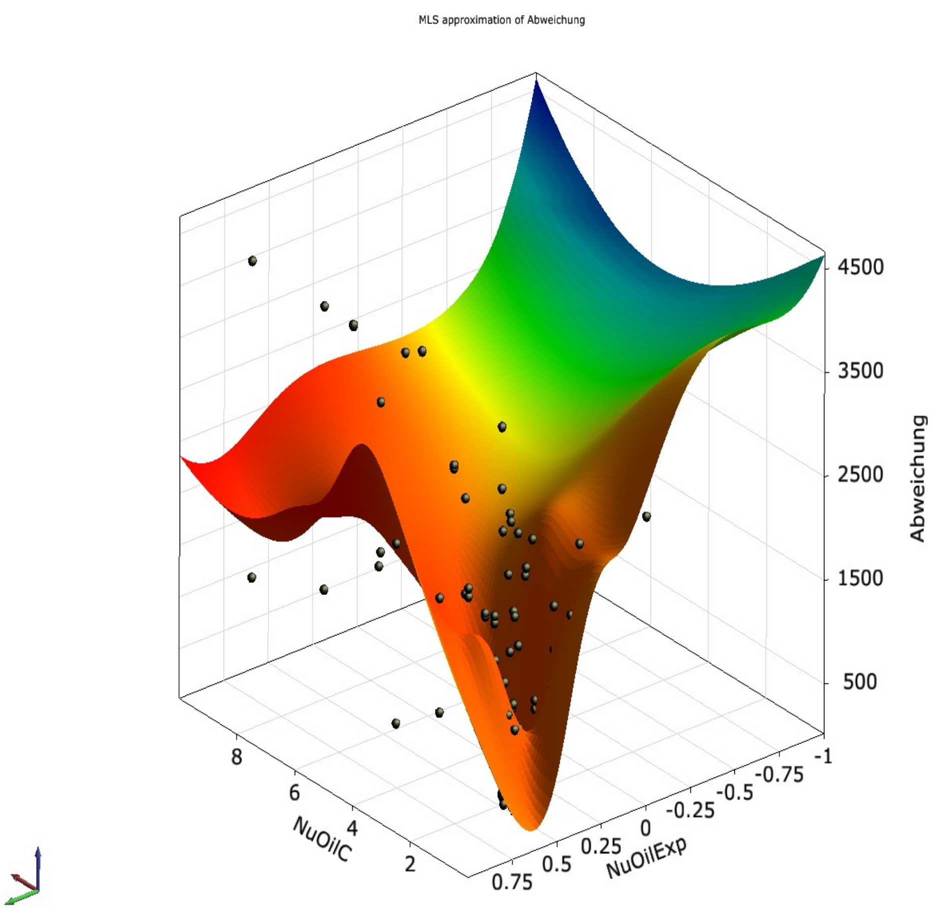
Select P_{t+1} from P_t;

Alter P_{t+1};

$t = t + 1$;

until t = maxTrials;

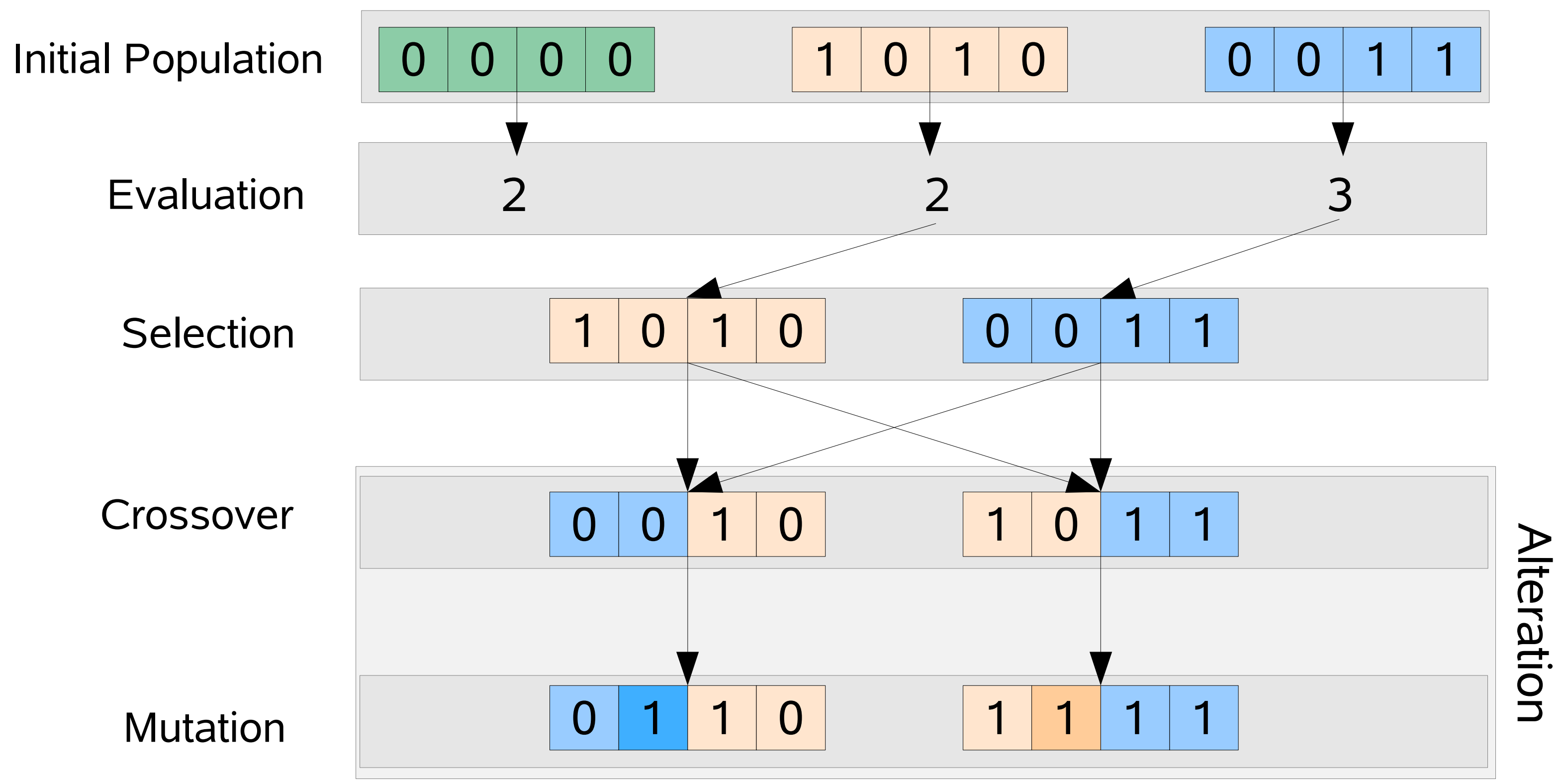
return best point in P_t;



Evolutionary Algorithms

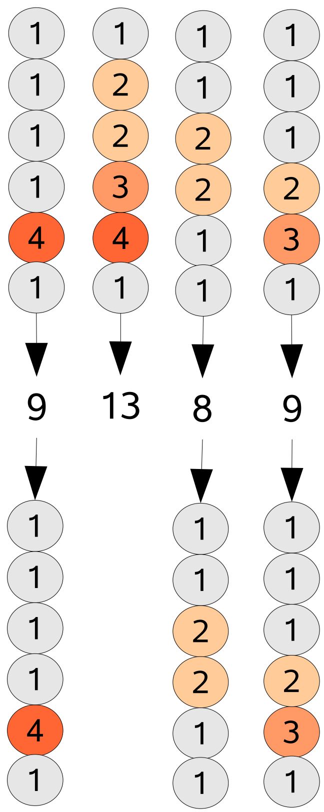
GA-SAT

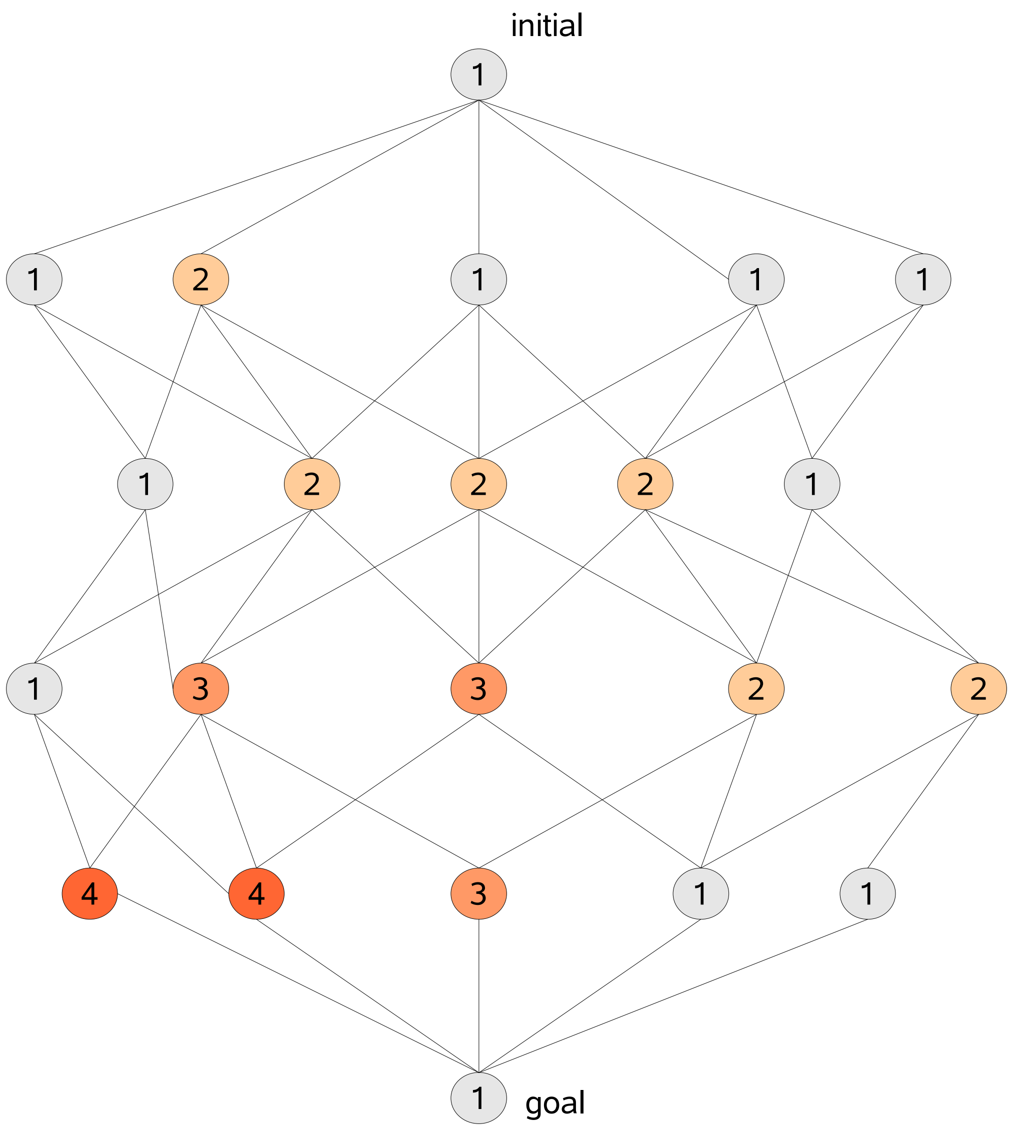
$$(x_1 \vee \overline{x}_2) \wedge (x_2 \vee \overline{x}_3) \wedge (x_2 \vee x_4) \wedge (x_1 \vee x_3)$$



Evolutionary Algorithms

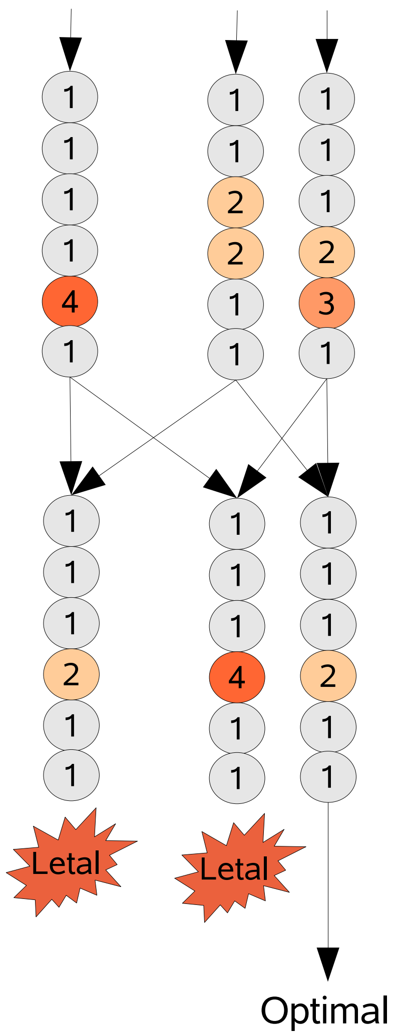
GA-PathFinding

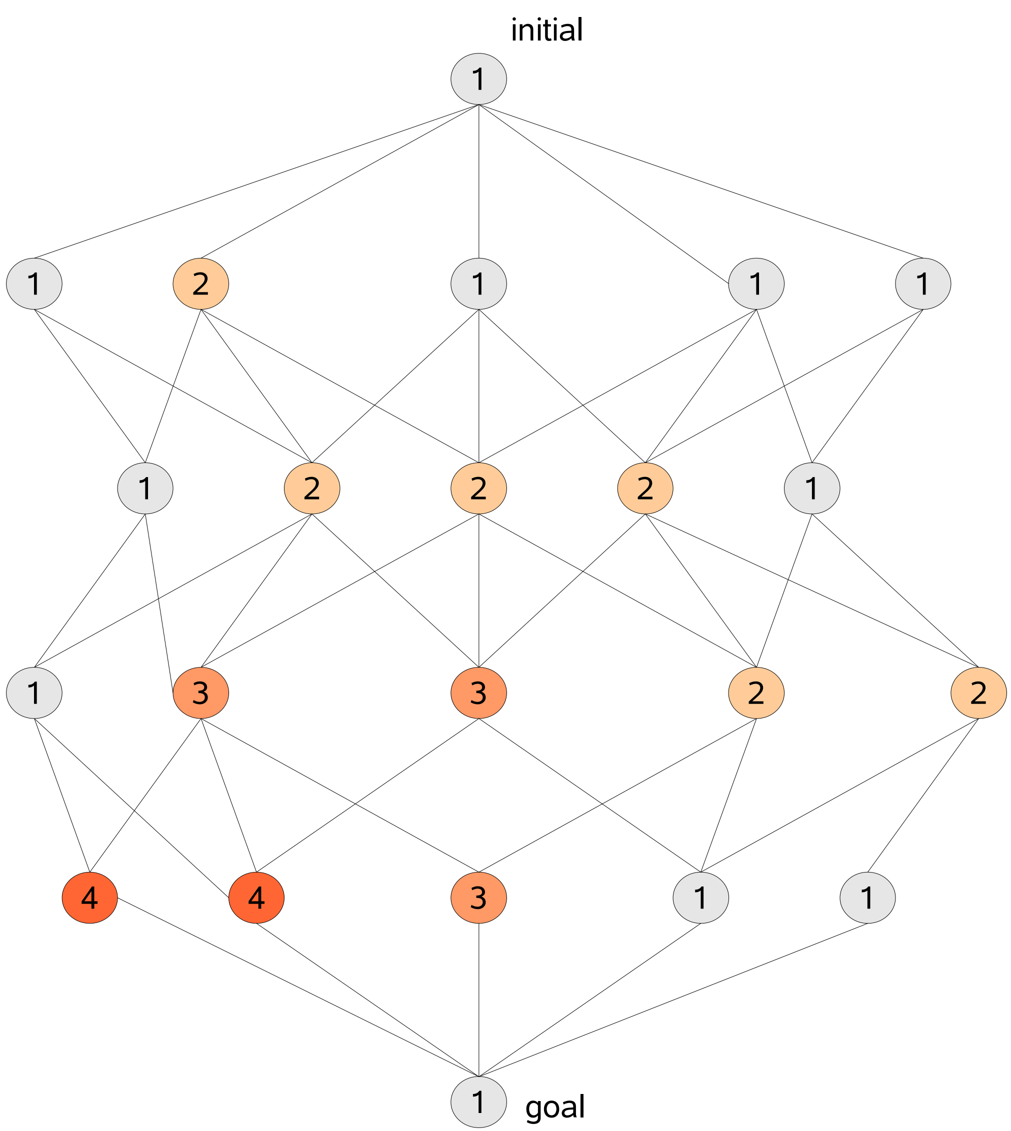




Evolutionary Algorithms

GA-PathFinding





Evolutionary Algorithms

GA-NLP

Initialization

Randomly choose a positive for x_i and use its invser for x_{i+1}. The last variable is either 0.75 (odd) or multiplied by 0.75 (even)

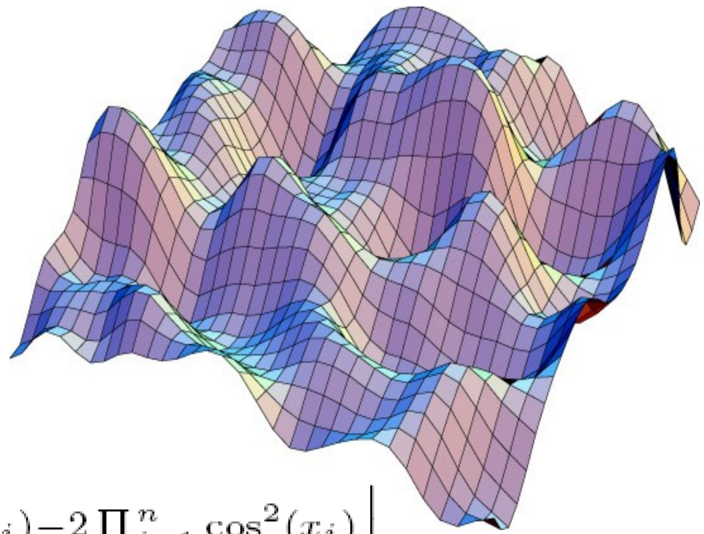
Crossover

$(x)(y) - (x^{\alpha} y^{1-\alpha})$

α randomly chosen in [0,1]

Mutation

Pick two variables randomly, multiply one by a random factor $q > 0$ and the other by $1/q$



Maximize

$$\left| \frac{\sum_{i=1}^{n} \cos^4(x_i) - 2\prod_{i=1}^{n} \cos^2(x_i)}{\sqrt{\sum_{i=1}^{n} i x_i^2}} \right|$$

Subject to

$$\prod_{i=1}^{n} x_i \geq 075, \ \sum_{i=1}^{n} \leq 7.5, \ 0 \leq x_i \leq 10$$

$N = 50$
population of size 30,
30000 générations,
probability of crossover 1
probability of mutation 0.06

Solution 0.833197
Better than any other algorithm !