(meta) heuristic algorithms

Laurent Lemarchand
Lab-STICC/UBO
Laurent.Lemarchand@univ-brest.fr

Combinatorial Optimization
Heuristic methods

- **Approximative** result, while a **part of random** but
  - Sometimes only available method (*e.g* program optimization)
  - Or exact methods for approximative model only (*e.g* circuit testing)
- Usefullness
  - Combinatorial explosion
  - Multiple or fuzzy objectives
  - Variability (robustness)
  - Fast runtimes more important than performance

Combinatorial Optimization
Greedy methods

- *Greedy* : build a solution step by step, they never come back on partial choices
  - Often far from optimality
- Very fast
  - Local search improvement possible
- Examples
  - Knapsack
  - Covering
  - Maximum stable
  - TSP

Greedy algorithms
Knapsack

- *knapsack* : fill a knapsack of limited weight, choosing the most interesting objects within a list
- Linear program
  - What mean variables
    $a_1x_1 + a_2x_2 \; \ldots \; + a_nx_n \leq b$
    $\mathit{max} \;\; c_1x_1 + c_2x_2 \; \ldots \; + c_nx_n$
    $x_j \leq \beta_j \qquad X_j \in \mathbb{N}$

- Order by profit $c_1 / a_1 \leq c_2 / a_2 \; \ldots \; \leq c_n / a_n$
- let $x_1 = \min(\beta_1, b/a_1)$; $b = b - a_1x_1$
- Iterate on $x_2, x_3, \ldots, x_n$

Greedy algorithms
Covering

- Cover n elements by at least one object, each object has its own cost. Minimize total cost of chosen objects
- A, covering matrix. What mean variables A_i^j

  $A_i^j \geq 1$

  min $c_1x_1 + c_2x_2 + \ldots + c_mx_m$

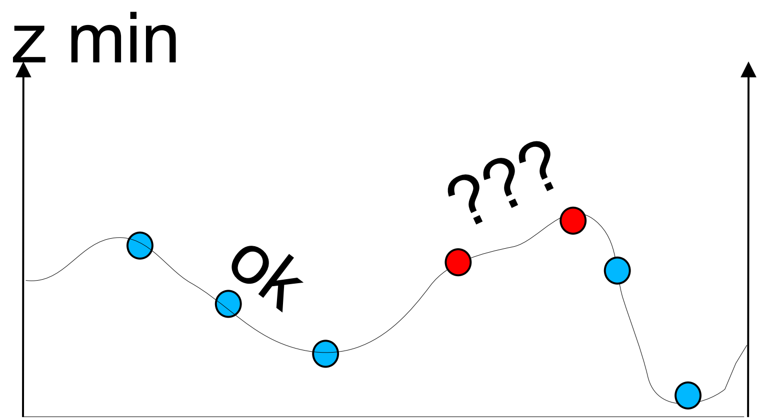
  $x_j \leq 1$ and $x_j \in \mathbb{N}$
- Find k s.t $c_k/a_j = \mathit{min}_{\ j \in 1..n}\ c_j/a_j$ with $a_j = \sum_{i=1..m} A_i^j$
- Delete column k and rows such that $A_i^k = 1$
- Iterate on reduced problem

Greedy algorithms
Travelling salesman problem

- Closest Neighbourgh algorithm
  - Random starting town
  - Go to next closest unvisited town
  - Loop from last to first town

Neighborhood methods (local search) principle

- Problem to face : go out local extrema
  - Random exploration, even costly
  - We have to focus on one solution (convergency)
- Single solution evolution
  - Descent method
  - Simulated annealing
  - Tabu search
- Or multiple solutions
  - Genetic algorithm
  - Ants



Local search
neighborhood

- Define a neighborhood function $V : S \rightarrow S^n$
  - Provide a set of n solutions similar (close to) $s \in S$
  - Explore these n solutions in order to find one that is better than s

- Sometimes non polynomial algorithm
- Optimality ?
- Ending criteria ?

- Alternatively, search from a set of seed solutions

Local search
descent

- Objective function *min* $f : S \rightarrow \mathbb{R}$
- Local search method $V : S \rightarrow S^n$
    - Provide a set of n solutions similar (close to) $s \in S$

generate an initial solution s_0

$s = s_0$

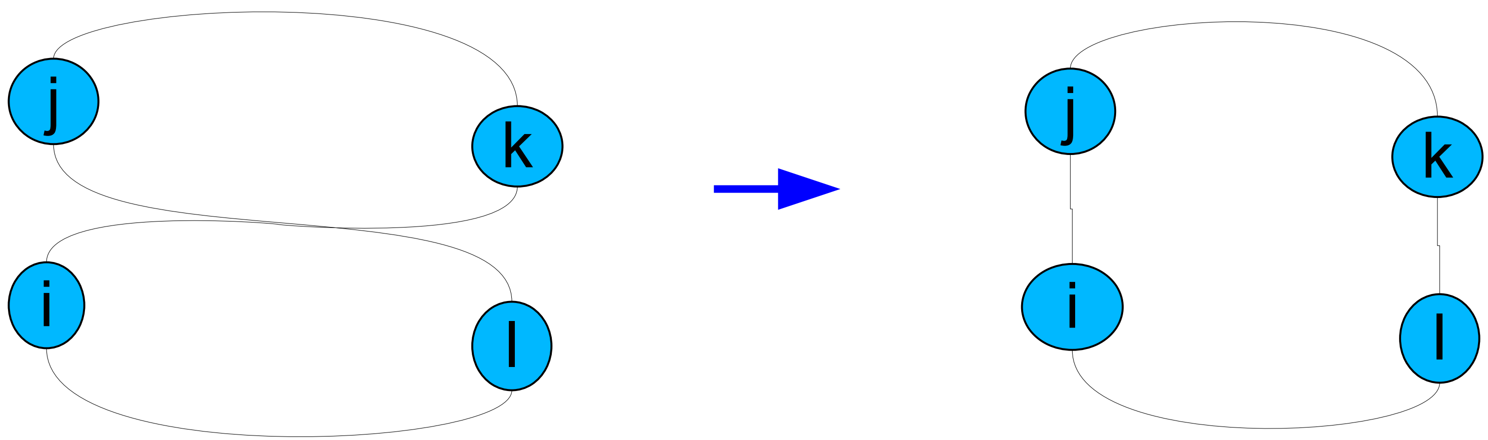
While end not reached

$s' = \min_{s \in V(s)} f(s)$

Local search
TSP Problem

- Example : *2-opt for* TSP local search(Lin, 1965, n(n - 3)/2)
  - Neighborhood of n^2 tours

    $T' = T \cup \{ i \rightarrow k, j \rightarrow l \} \setminus \{ i \rightarrow j, k \rightarrow l \}$

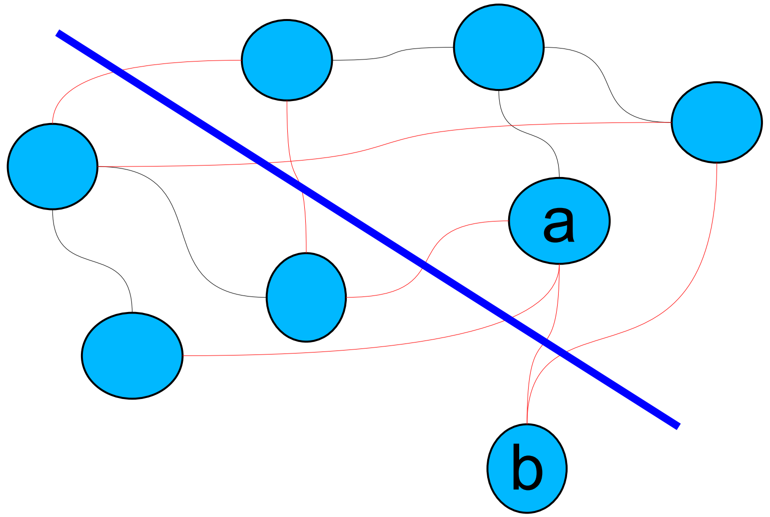


- *3-opt* possible, but very large neighborhood n(n – 3)(n - 2)

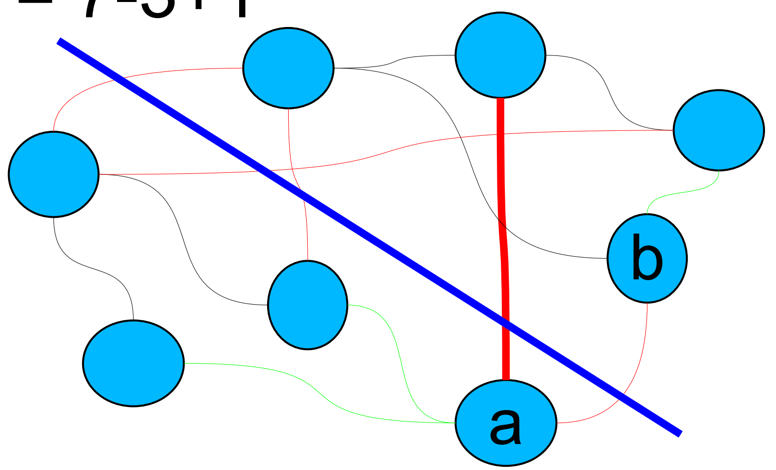
Local search
2-way Partitioning problem

- Graph G = (V, E) avec |X| = 2n find a partition $X = V_1 \cup V_2$ t.q $|V_1| = |V_2| = n$ which minimizes the number of edges crossing the 2 parts
  - Pairwise exchange neighborhood

c=7

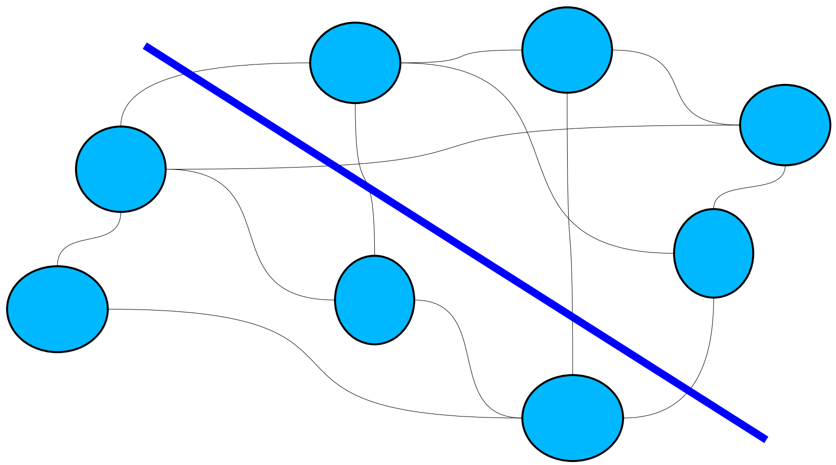


c=5 = 7-3+1



2-way partitionning
Kernighan-Lin heuristic

- At each step, choice the exchange maximizing the cut number gain
  - Constraint : a node can be swapped only one time
  - N/2 steps at most



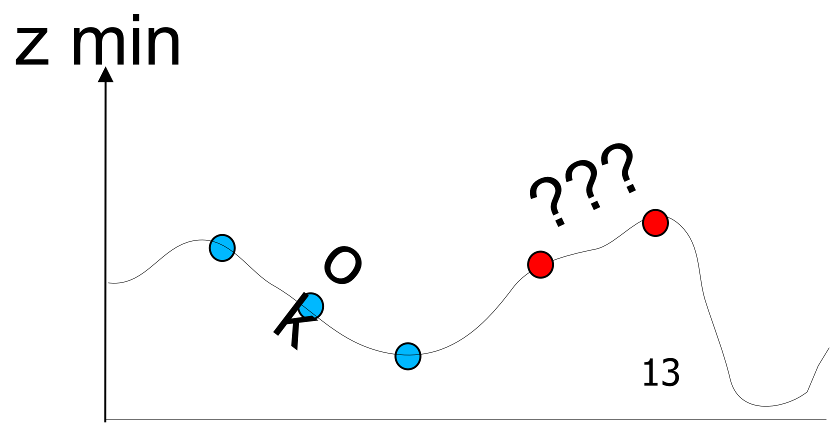
Improving descent methods
remarks

- Recall : trade off to be found between
  - Improving current solution
  - Performing an efficient search of *all the search space*

Exploitation ⟷ Exploration

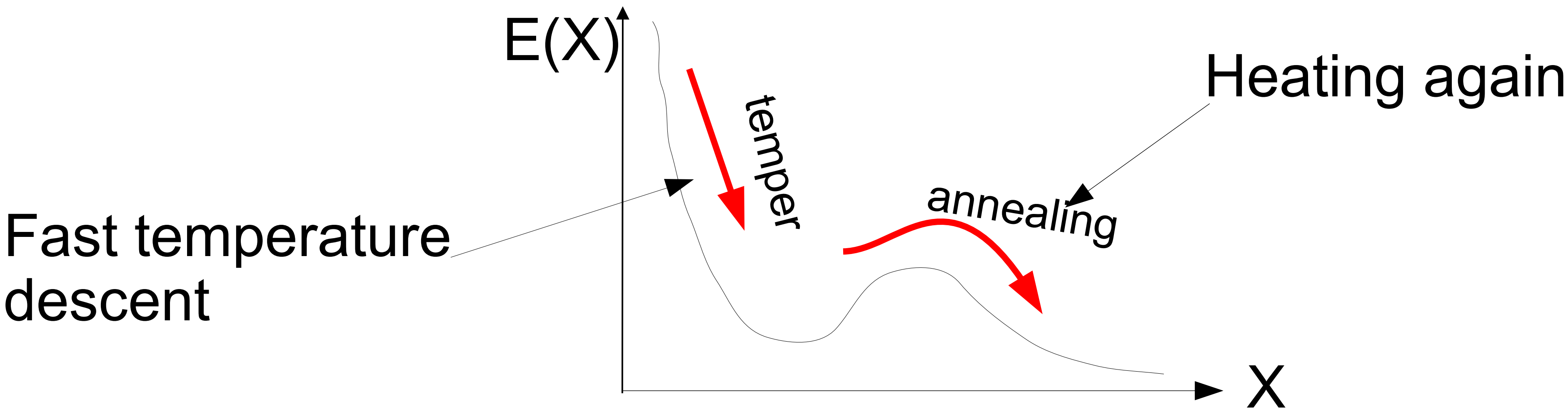
- Always the local extrema problem

- Tradeoff



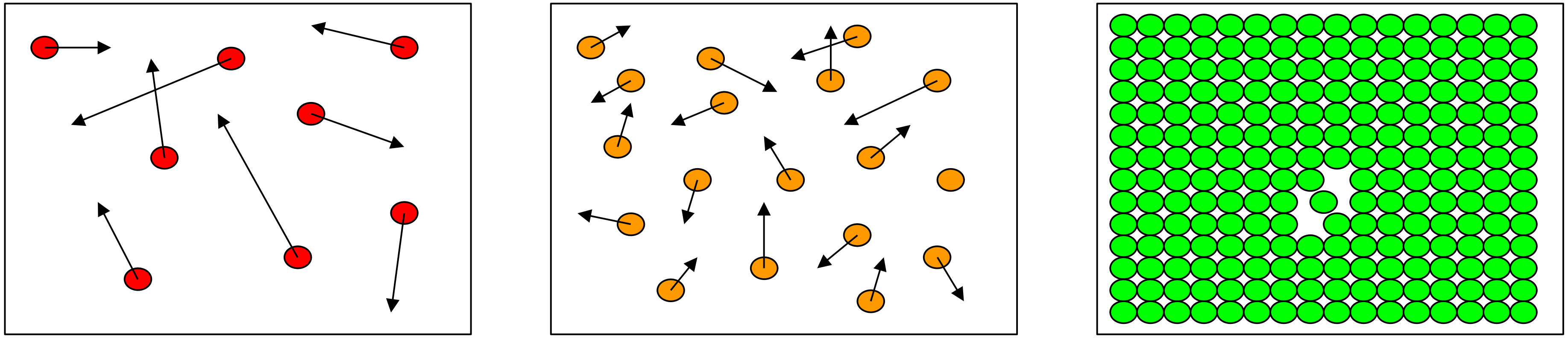
Meta heuristics
simulated annealing (SA)

- Applicable to a lot of problems
  - Dynamic evolution of the tradeoff exploit./explor.
  - Based on crystal metal cooking model
- Evolution of energy levels within the metal

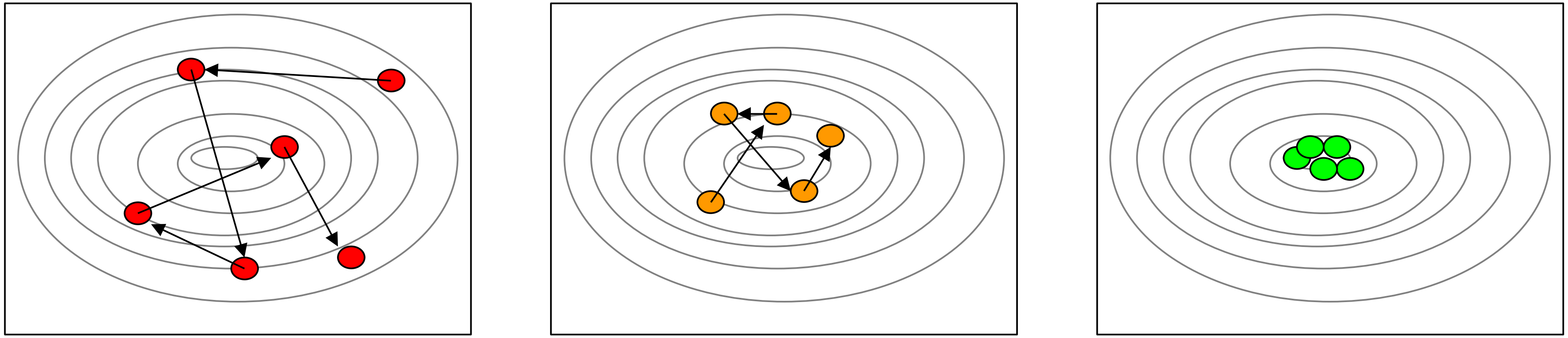


Simulated annealing
analogy physics/optimization

physics



optimization



Simulated annealing principle

- First Explore randomly the search space
- Vary the degree of non determism (temperature)
  - Start high level : **exploration**, with a very random behavior
  - End low level : **exploitation** as with descent methods

Simulated annealing
one step

- From current solution s_i, explore its neighborhood to obtain s_{i+1}
- If $f(s_{i+1}) - f(s_i) < 0$, accept s_{i+1} (*minimisation*)
- Else, accept s_{i+1} based on probality :

$$p(s_i \rightarrow s_{i+1}) = \mathrm{e}^{\frac{-(\ f(s_{i+i}) - f(s_i)\ )}{T}}$$

≈Gibbs-Bolzmann distribution

Simulated annealing algorithm

Space S,

Evaluation f(s)(*min*),

temperature(T)

accept(Δf,T)

Initial temperature

$T = T_0$

Best solution

$s = s_{best}$ = greedy()

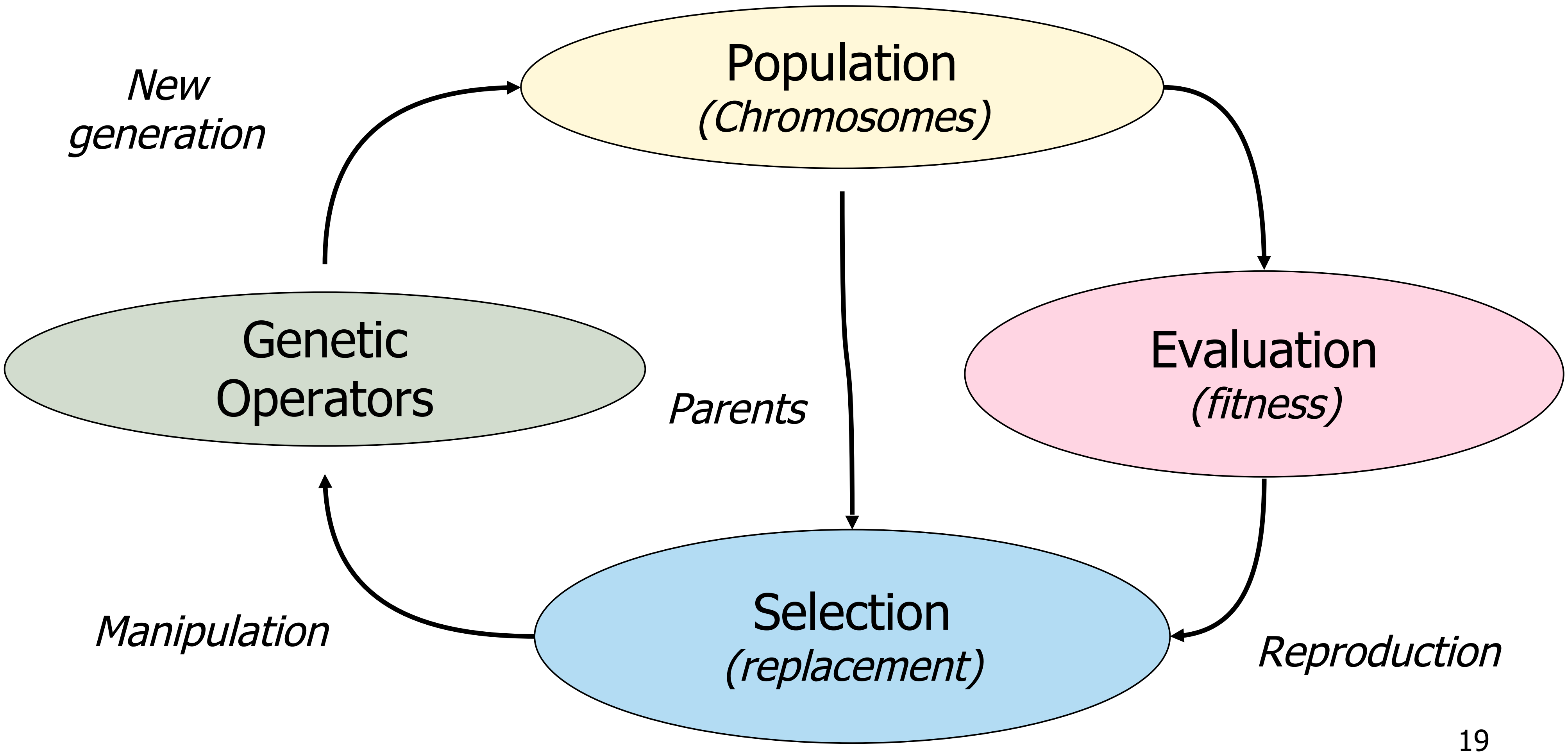
Algo SA
while criteria1 **do**

  while criteria2 **do**
    s_2 = neighbor(s, S)
    $\Delta f = f(s_2) - f(s)$
    if $f(s_2) < f(s_{best})$
      $s_{best} = s_2$
    if $\Delta f < 0$ **or** accept(Δf, T)
      $s = s_2$
  endw

  T = temperature(T)
Endw
End SA

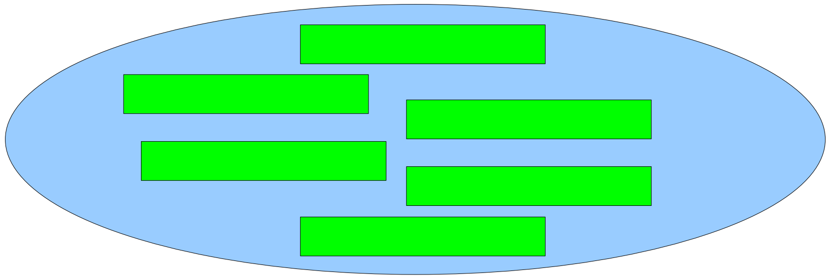
Genetic algorithms
Evolutionary algorithms

Overview

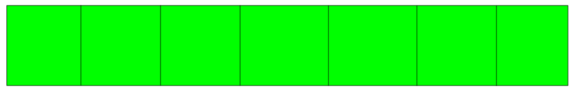


Genetic algorithms
Data

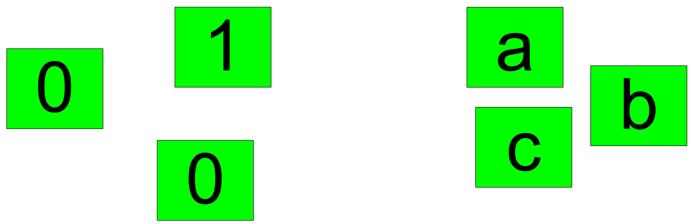
- Population
  - Set of individuals



- Individual
  - Encode a solution
  - Chromosome based representation

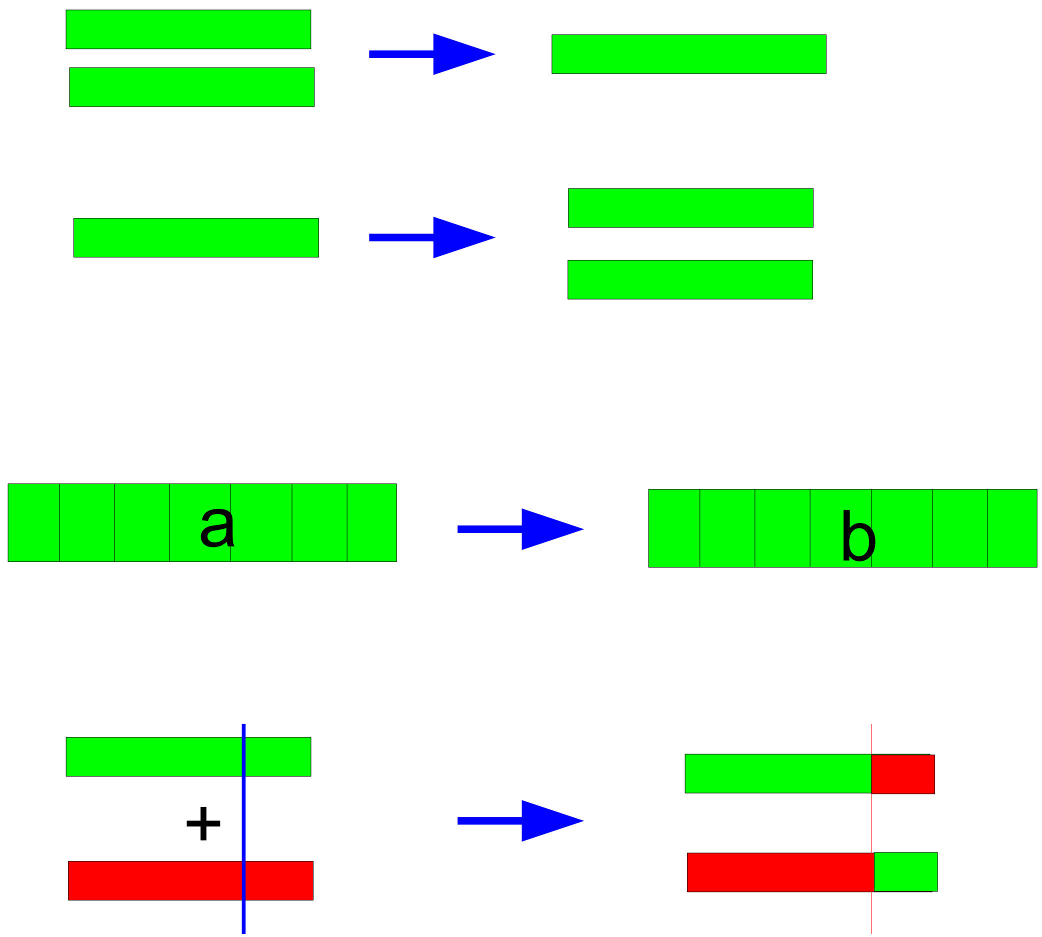


- Chromosomes
  - Allele based



Genetic processing
based on chromosomes

- Population
  - Reproduction
  - Duplication
- Individual
  - Mutation
  - Crossover



Genetic algorithms
algorithm

Space S,

Evaluation f(s)(*min*),

crossover(P)

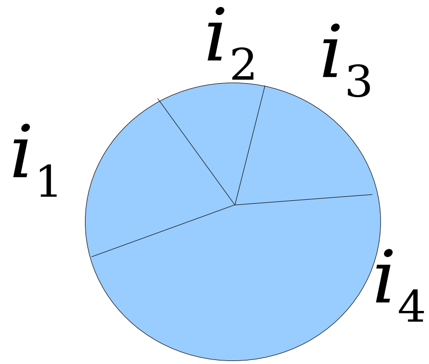
mutation(P)

Mutation and crossover rates

Algo GA
generate initial population P in S
while end not reached **do**
    reproduction(P) according to f()
    crossover(P)
    mutation(P)
endw
Best solution within population
end GA

Genetic processing
ramdom choices

Exploitation
of
the search space



- Population
  - For reproduction (*roulette wheel*)

    Chances to be selected proportional to fitness
- Operations
  - Mutation rate
    example: 0,5%
  - Crossing over

    % of population

    Random cutting sites

Exploration
of
the search space

Genetic algorithms implementation

- Empirical technique
- Lot of freedom for
  - Chromosome coding (*building blocks*)
  - Probabilities
  - Operators
  - Coupling witch others approaches
- Parallelism

Genetic algorithms parallelism

- Different alternatives
  - Master/slave oriented, fine-grain model (// based on individual evaluation)
    - + Local search applicable
  - Large grain approach (// based on the evolution of multiple independant populations)
    - Island model (with some individuals migrating from populations)

Genetic algorithms
parallelism - example

- Benchmark TSPLIB
  http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/index.html
  - 2D instances (euclidian distances)
  - Island model, 10 stations, ethernet 100Mb/s,
    MPI (// interprocs)
    - 1000 generations
    - 1000 individuals
    - Mutation 30%, (cross over ?)
    - Circular migration, 100 iterations, 20% of population.
    - 1 point Crossover, roulette whell selection

Genetic algorithms parallelism - example

Comparison of Parallel Metaheuristics for Solving the TSP
M. Lazarova, P. Borovska
CompSysTech’08

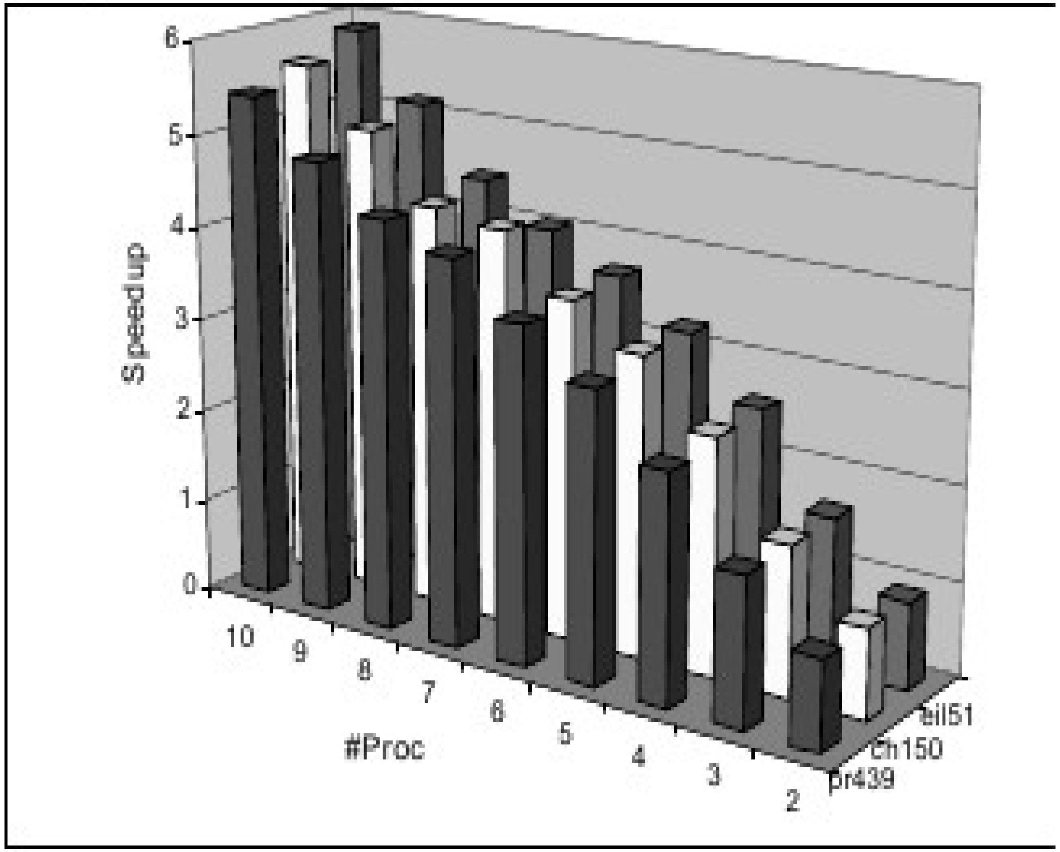


Fig.6. Speedup of parallel GA