Parallel and distributed evolutionary algorithms

Prof. El-Ghazali TALBI

University Lille 1

El-ghazali.talbi@univ-lille1.fr

www.lifl.fr/~talbi

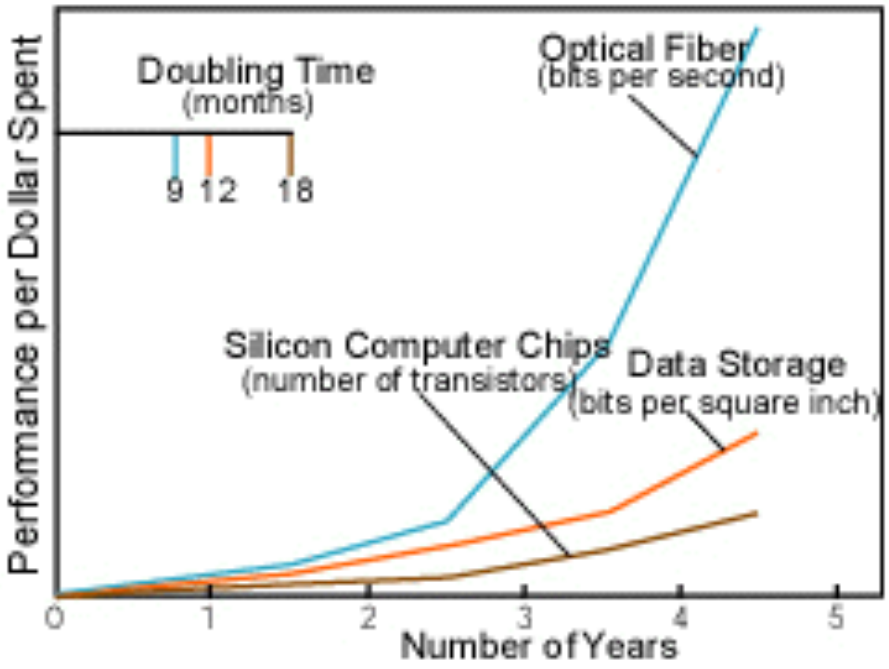
Motivations

- High-dimensional and complex optimization problems in science and industry
  → networks, genomics, transportation, engineering design, power systems, …

- Metaheuristics (ex. EAs) reduce the computational complexity of search BUT Parallel issue remains important:
  - More and more complex metaheuristics
  - Huge search space
  - CPU or memory intensive of the objective function

Motivations

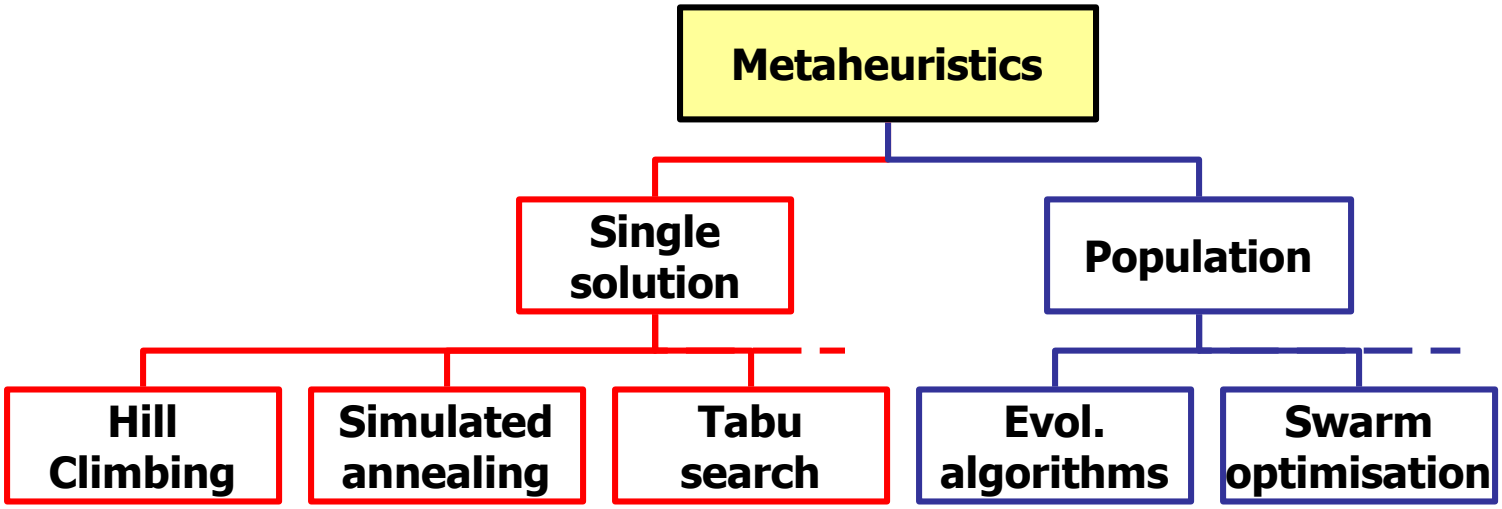
- Rapid development of technology
  - Processors: Multi-core processors, GPU, …
  - Networks (LAN & WAN): Myrinet, Infiniband, Optical networks, …
  - Data storage
- Increasing ratio performance / cost



Goals

- Speedup the search → real-time and interactive optimization methods, dynamic optimisation, …
- Improve quality of solutions → cooperation, better even on a single processor
- Improve robustness (different instances, design space)
- Solving large problems → instances, accuracy of mathematics models

Taxonomy of metaheuristics



- Solution-based metaheuristics: HillClimbing, Simulated Annealing, Tabu Search, VNS, …
- Population-based metaheuristics: Evolutionary Algorithms, Scatter Search, Swarm optimization, …

Outline

- **Parallel Metaheuristics: Design issues**
- Parallel Metaheuristics: Implementation issues
  - Hardware platforms, Programming models
  - Performance evaluation
- Adaptation to Multi-objective Optimization
- Software frameworks
- Illustration : Network design problem

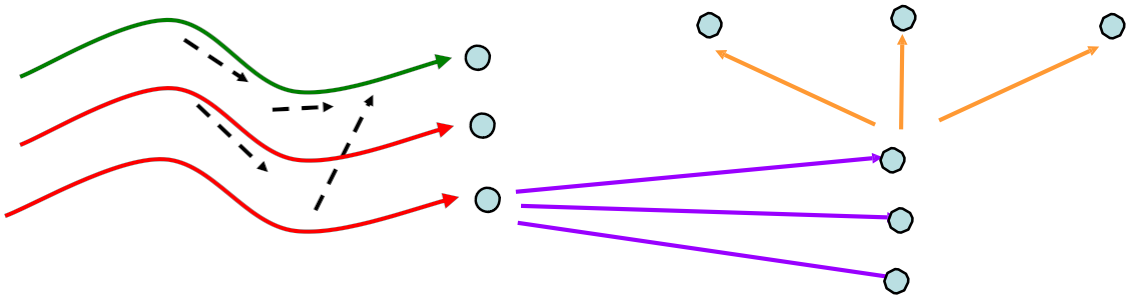
Parallel Models for Metaheuristics

- A unified view for single-based metaheuristics and population based metaheuristics
- Three major hierarchical models:
  - **Algorithm-Level**: Independent/Cooperative self-contained metaheuristics
  - **Iteration-Level**: parallelization of a single step of the metaheuristic (based on distribution of the handled solutions)
  - **Solution-Level**: parallelization of the processing of a single solution

Parallel Models for Metaheuristics

Solution-level:
Processing of a single solution
(Objective / Data partitioning)

Algorithm-level:
Independent walks,
Multi-start model,
Hybridization/Cooperation
of metaheuristics



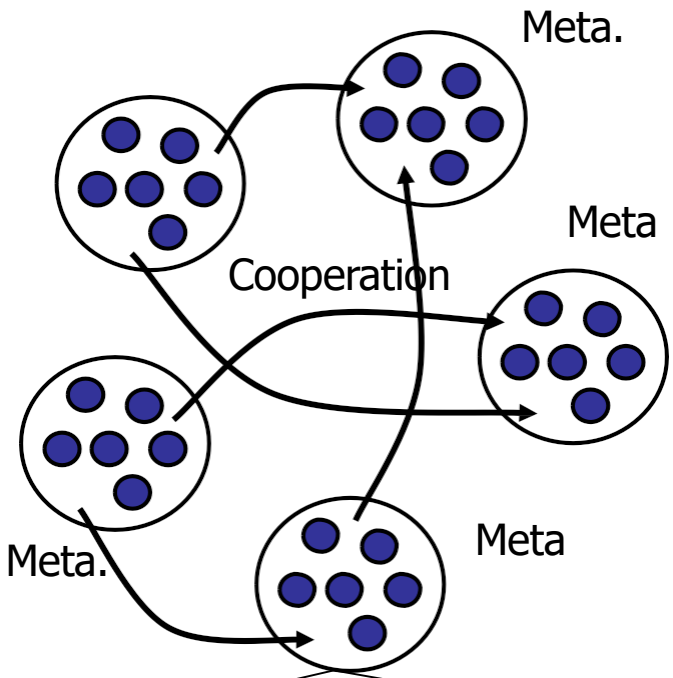
Iteration-level:
Parallel evaluation of
the neighborhood/population

Scalability $|H| \times |P| \times |S|$

Search algorithms — Population / Neighborhood — sub-solutions

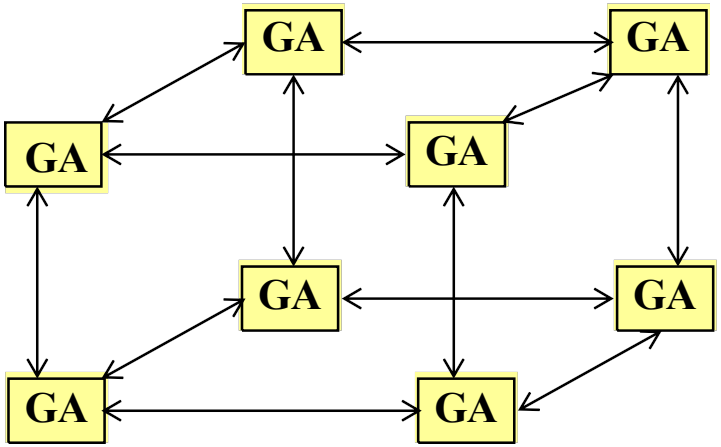
Algorithm-level Parallel Model

- Problem Independent
- Alter the behavior of the metaheuristic
- Design questions:
  - When ? Migration decision:
    - Blind: Periodic or Probabilistic
    - Adaptive: state dependent
  - Where ? Exchange topology: complete graph, ring, torus, random, …
  - Which ? Information : elite solutions, search memory, …
  - How ? Integration



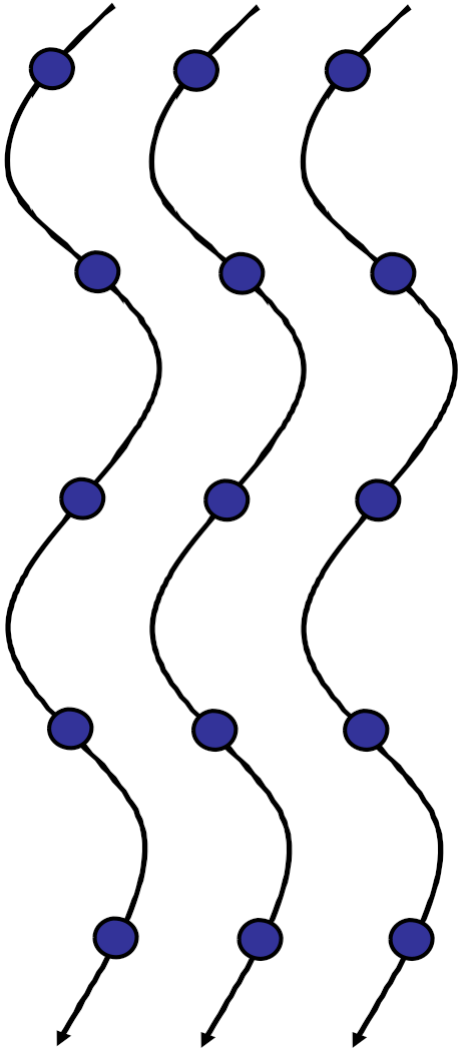
Simulated Annealing
Genetic Programming
Evolution Strategy
Tabu search
Ant Colonies
Scatter search, …

Ex : Evolutionary Algorithms (Island Model



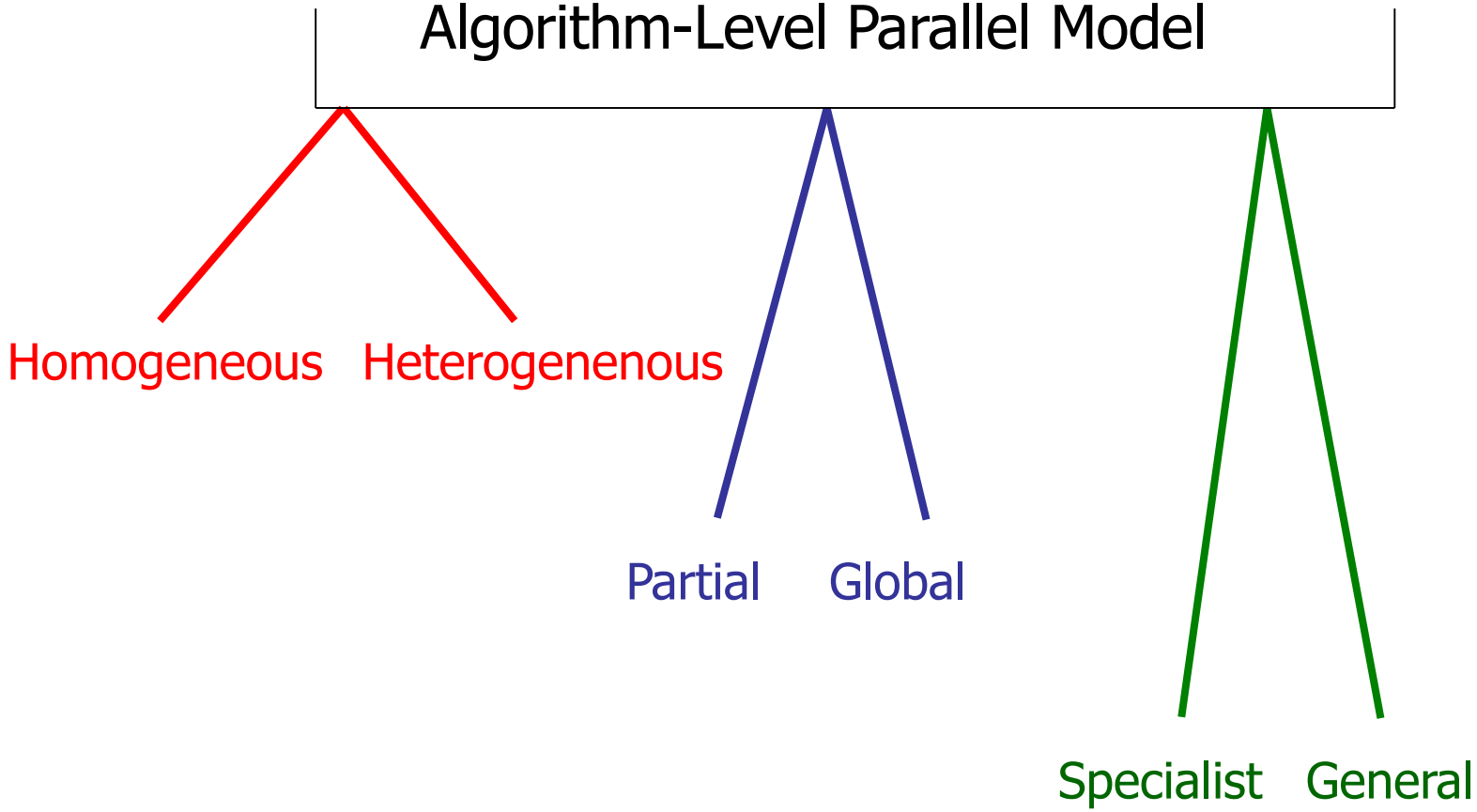
- Distribution of the population in a set of islands in which semi-isolated EAs are executed
- Sparse individual exchanges are performed among these islands with the goal of introducing more diversity into the target populations
- Improvement of robustness and quality

Ex : The multi-start model in Local Search



- Independent set of Local Searches
- Improve robustness and quality of solutions
- May start from the same or different initial solution, population
- Different parameters
  - encoding, operators, memory sizes (tabu list, …), etc.

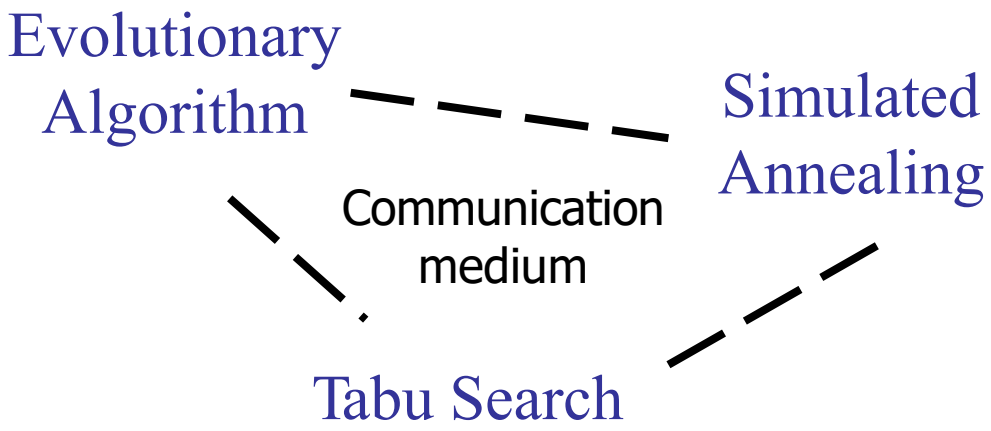
Flat classification



Homogeneous / Heterogeneous

- Homogeneous hybrids = Same Metaheuristic
- Heterogeneous = Different Metaheuristics

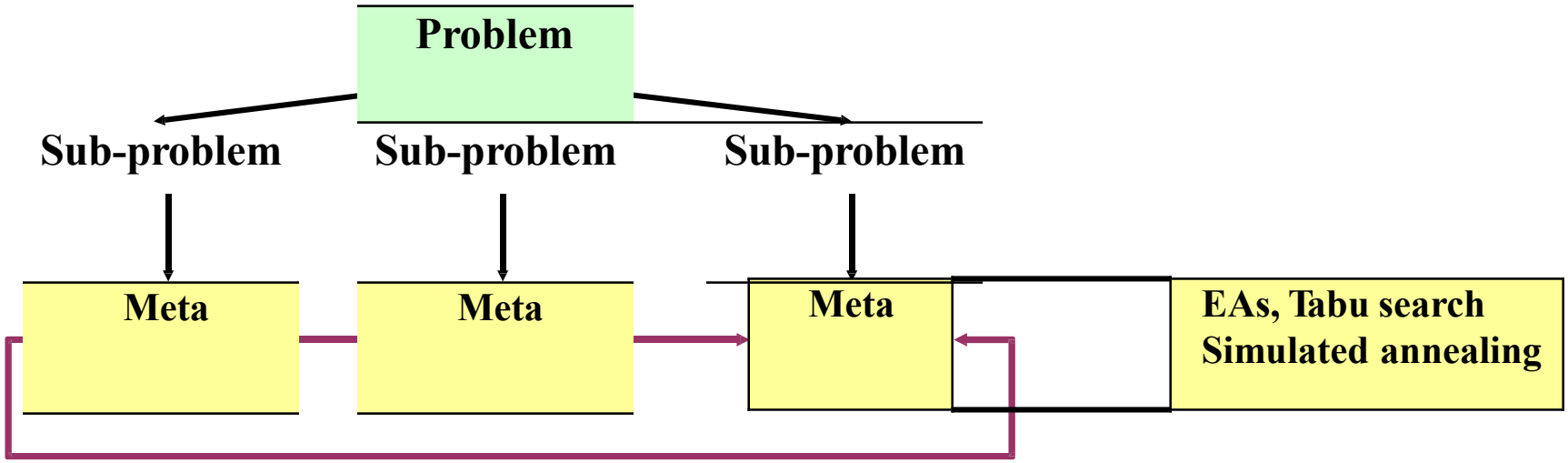
- Example



Several different metaheuristics cooperate and co-evolve some solutions

Global / Partial

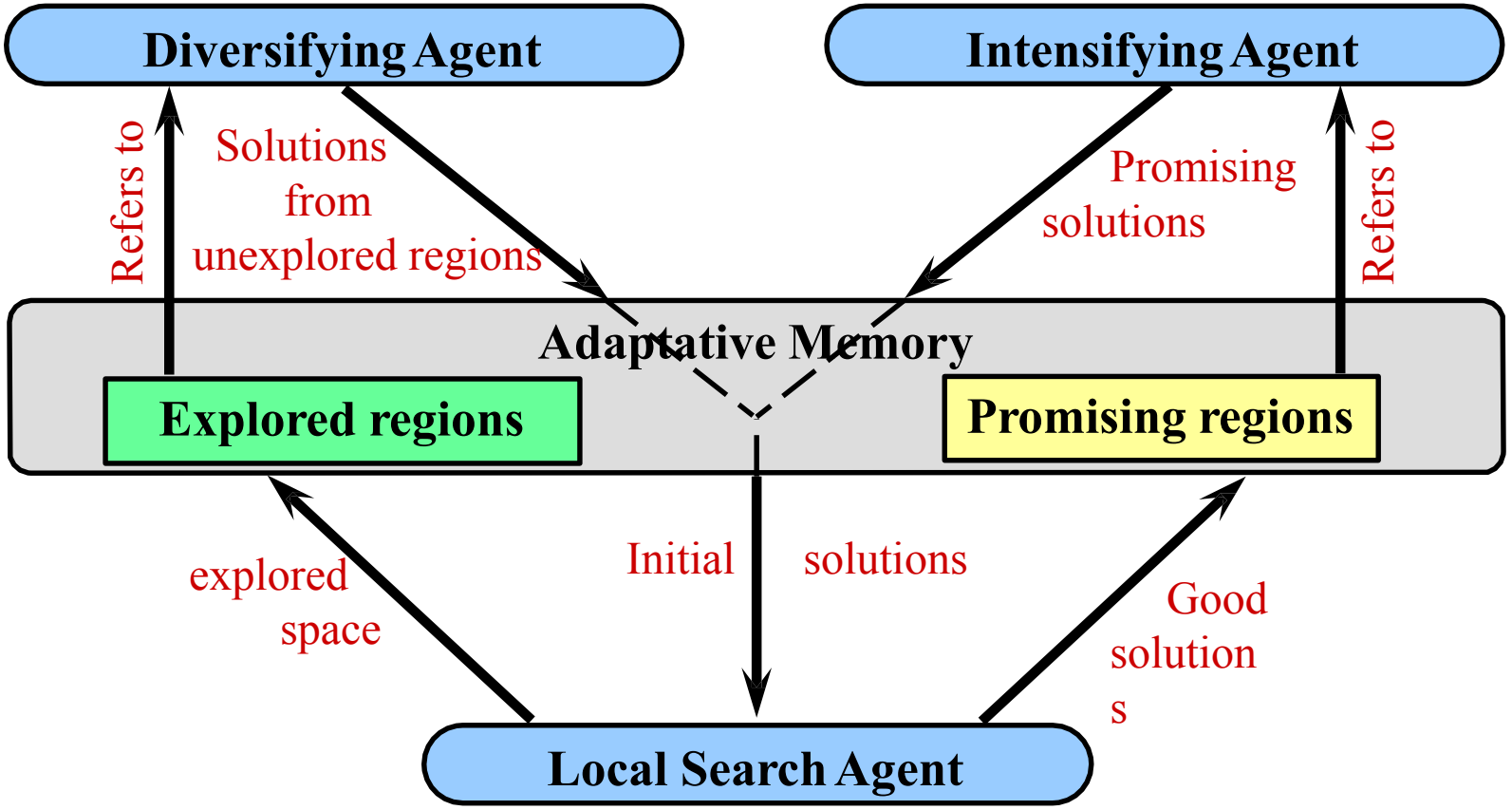
- Partial : Problem is decomposed in sub-problems. Each algorithm is dedicated to solve one sub-problem
- Combinatorial / Continuous : Decomposition of the problem, decision space, …
- Problem dependant: Vehicle routing, Scheduling, …



Synchronization : Build a global viable solution

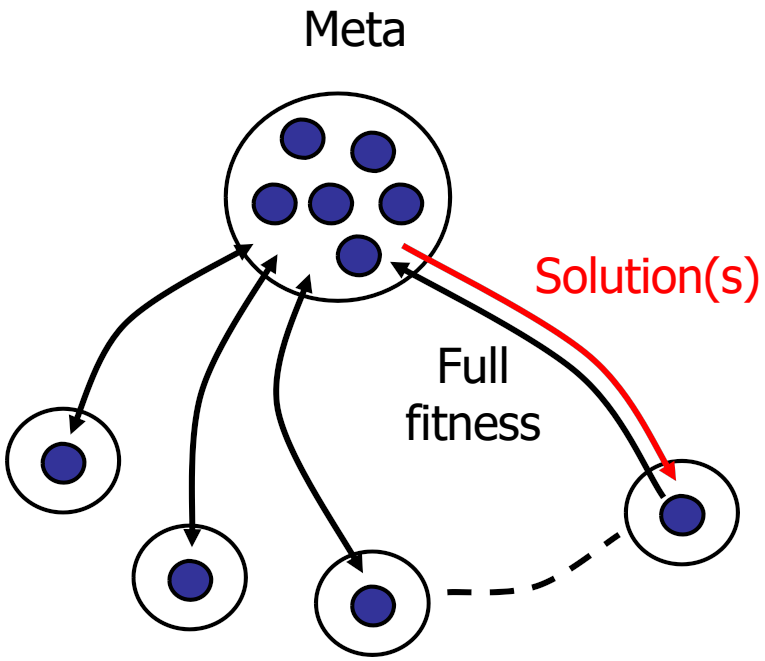
Specialist / General

- **Specialist**: Algorithms solve different problems (ex: Co-evolutionary)



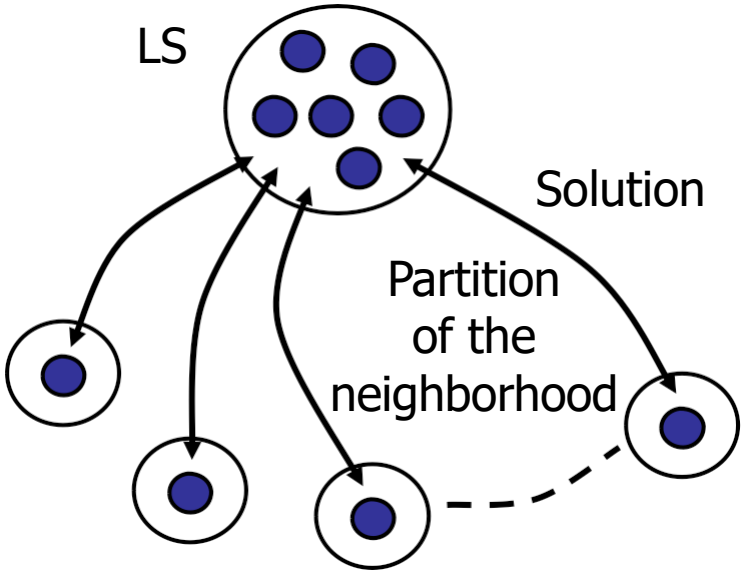
Iteration-level Parallel Model

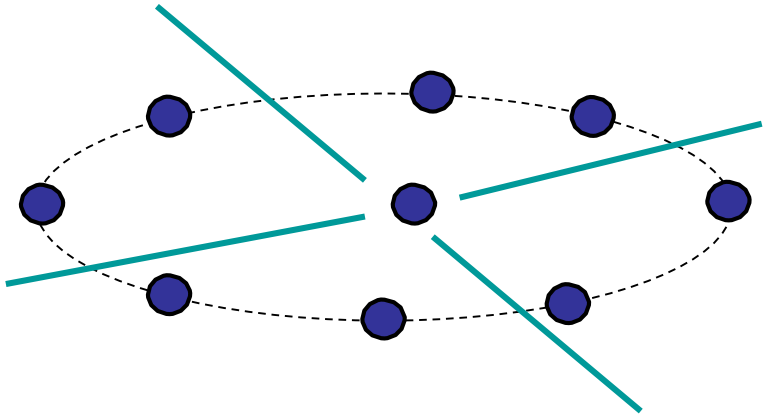
- Problem independent
- Complex objective functions (non linear, simulations, …)
- Do not alter the behavior of the metaheuristic → Speedup the search



- Design questions:
  - Single-solution based algorithms: decomposition of the neighborhood
  - Population based algorithms: decomposition of the population

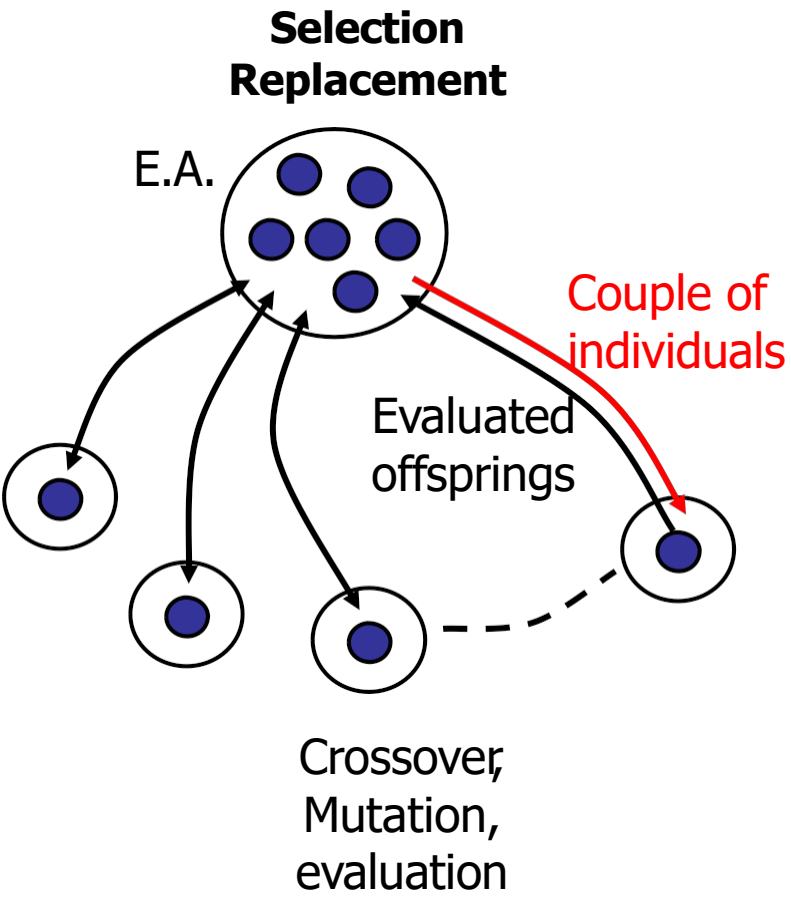
Ex : Single-Solution Based Metaheuristic





- Evaluation of the neighborhood: computationally intensive step
- Decomposition of the neighborhood :
  - Asynchronous for Simulated Annealing (# behavior)
  - Synchronous for deterministic algorithms (e.g. Tabu Search)

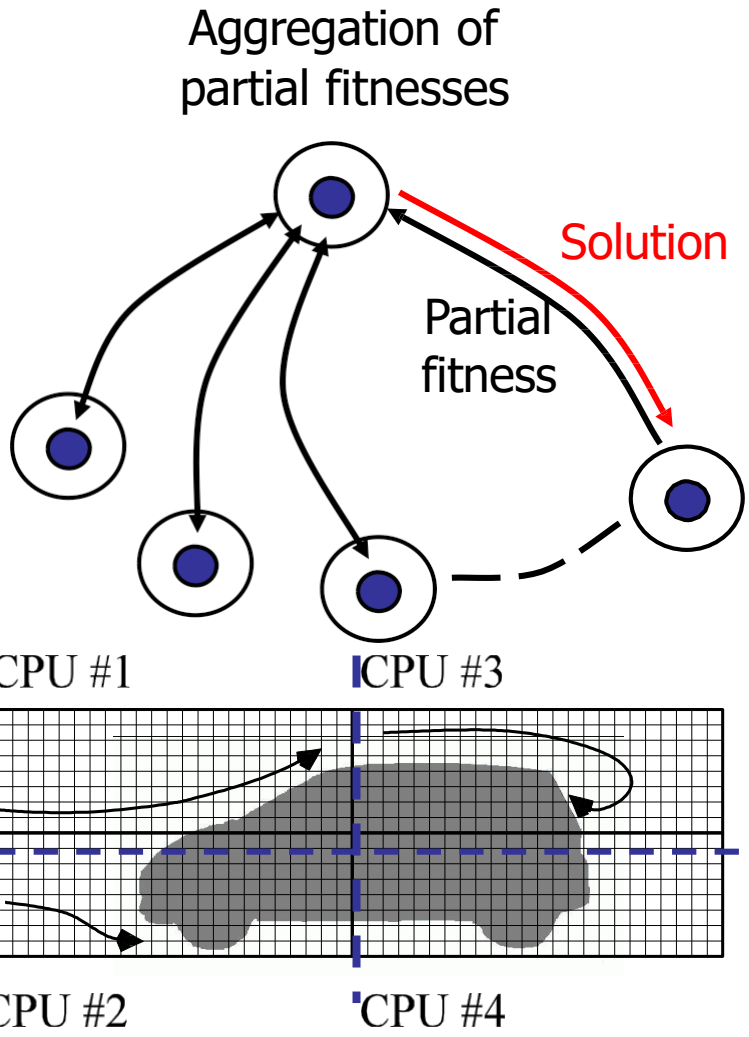
Ex : Population Based Metaheuristic



- Decomposition of the population:
  - Individuals (EAs), Ants (AS), Particles (PSO), etc.
  - Sharing Information: Pheromone matrix, etc.
  - Asynchronous for steady state GAs (# behavior)
  - Synchronous for generational GAs

Solution-level Parallel Model

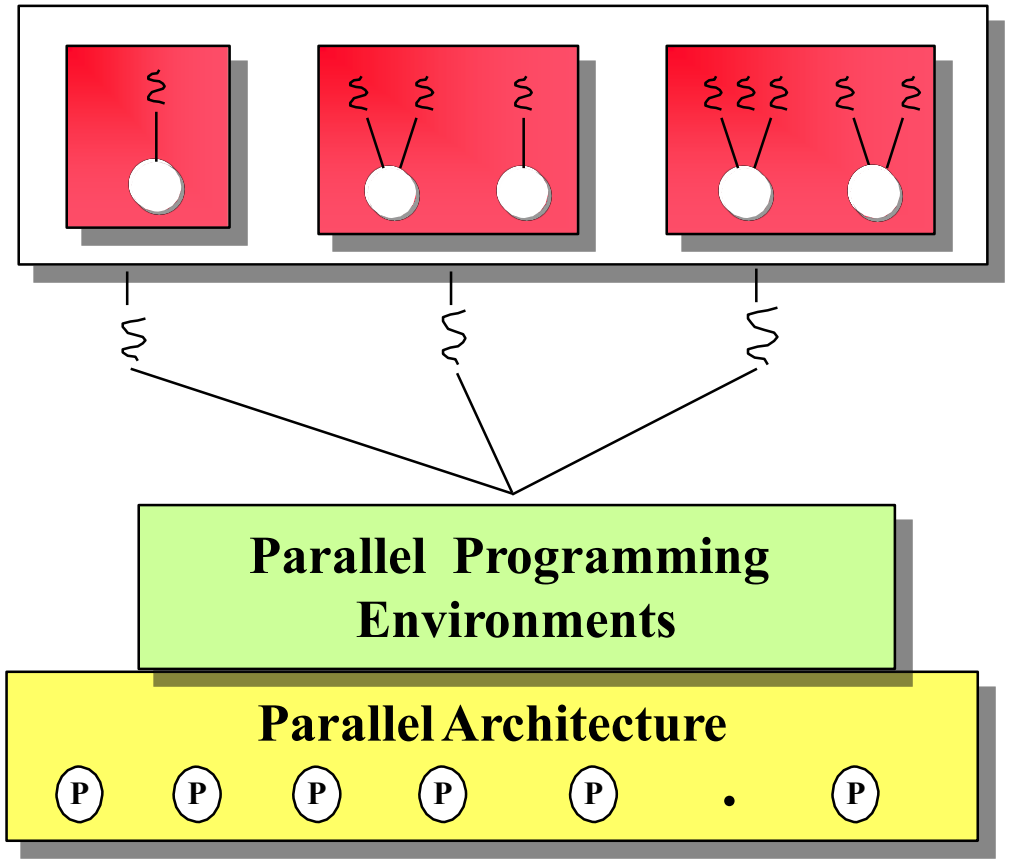
- Problem dependent
- Do alter the behavior of the metaheuristic → Speedup the search (CPU or I/O intensive)
- Synchronous
- Design questions:
  - Data / Task Decomposition
    - Data : database, geographical area, structure, …
    - Task : sub-functions, solvers, …



Outline

- Parallel Metaheuristics: Design issues
- **Parallel Metaheuristics: Implementation issues**
  - **Hardware platforms, Programming models**
  - **Performance evaluation**
- Adaptation to Multi-objective Optimization
- Software frameworks
- Illustration : Network design problem

Parallel Metaheuristics: Implementation issues



Design of Parallel Metaheuristics

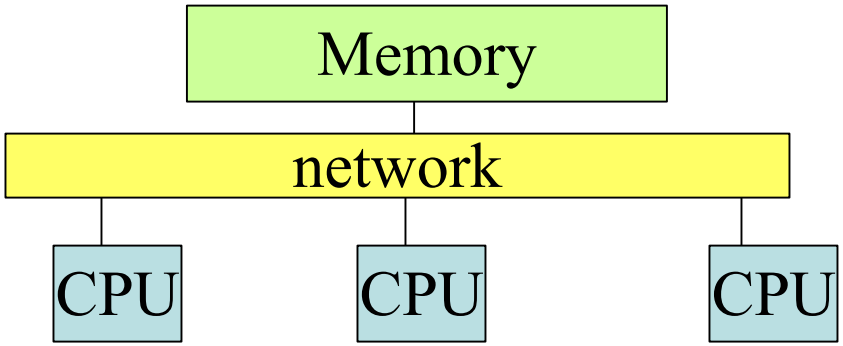
Programming Paradigms

Execution Support

Hardware

Main criteria : Memory sharing, Homogeneity, Dedicated, Scalability, Volatility

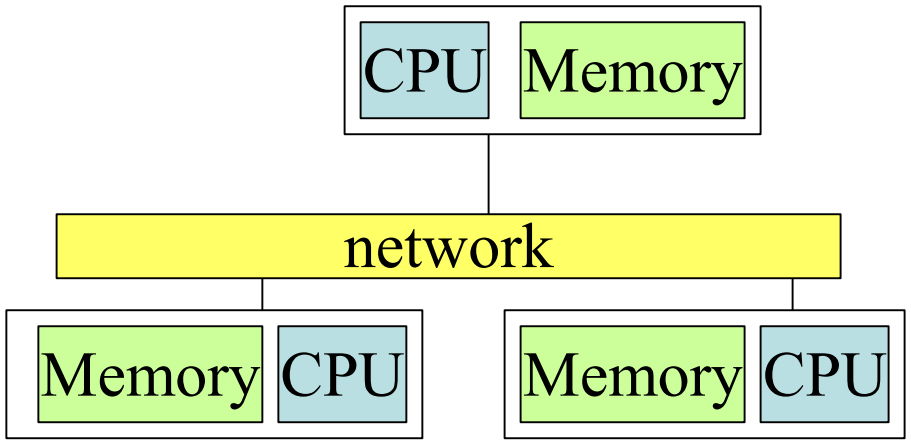
Shared Memory Machine (SMP)



interconnection network: bus, crossbar, multistage crossbar

- Easy to program: conventional OS and programming paradigms
- Poor Scalability: Increase in processors leads memory contention
- Ex. : Multi-core (Intel, AMD), Origin (Silicon Graphics), ...

Distributed Memory Architectures



Interconnection schema : hypercube, (2D or 3D) torus, fat-tree, multistage crossbars

- Good Scalability : Several hundred nodes with a high speed interconnection network/switch
- Harder to program
- Communication cost
- Ex. : Clusters

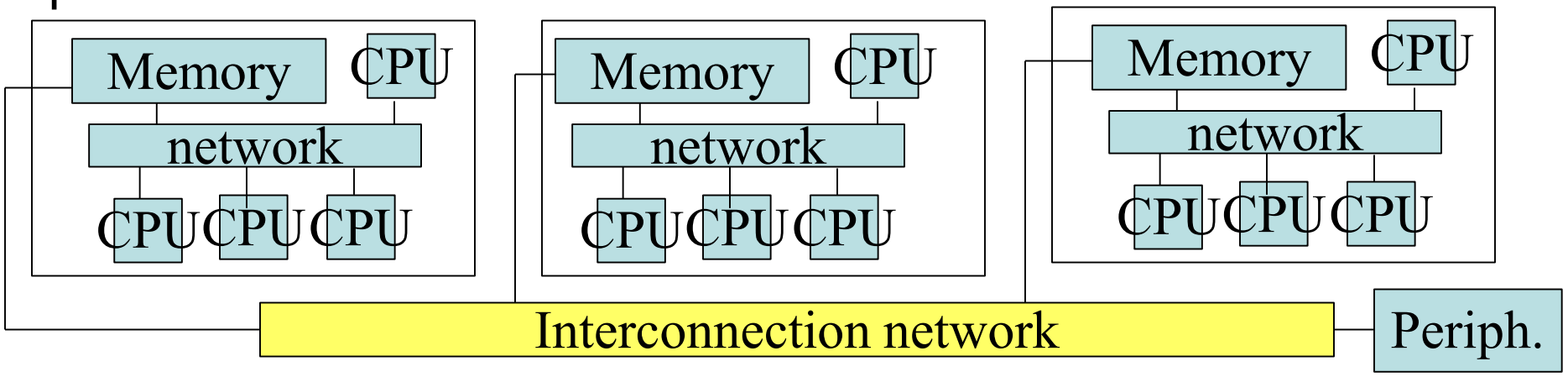
Clusters & NOWs (DSM)

- **Clusters**: A collections of PCs interconnected through high speed network,
  - Low cost
  - Standard components

- **NOWs**: Network Of Workstations
  - take advantage of unused computing power

ccNUMA architectures

- Mixing SMP and DSM
- Small number of processors (up to 16) clustered in SMP nodes (fast connection crossbar for instance)
- SMPs are connected through a less costly network with poorer performance



Grid Computing

"Coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations"

High-Performance Computing GRID

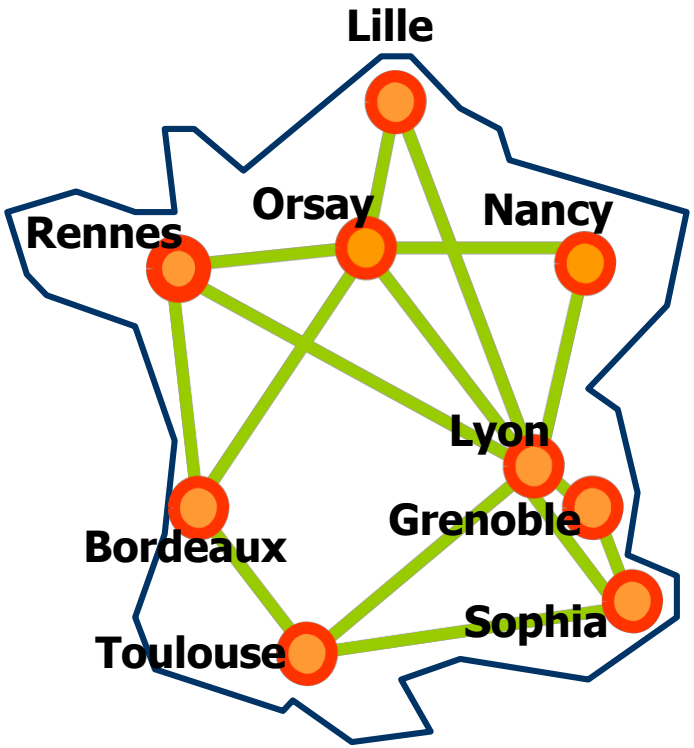
- Offer a virtual supercomputer

High-Throughput Computing GRID

- Billions of idle PCs …
- Stealing unused CPU cycles of processors (a mean of 47%)

- Inexpensive, potentially very powerful but more difficult to program than traditional parallel computers

GRID Platforms



HPC Grid: GRID'5000: 9 sites distributed in France and inter-connected by Renater
7800 proc: between 500 and 1500 CPUs on each site

HTC Grid: PlanetLab: 711 nodes on 338 sites over 25 countries

Main criteria for an efficient implementation

Criteria / Architecture	Shared Memory / Distributed	Homogenous / Heterogenous	Dedicated / Non Dedi.	Local network / Large network	Volatility & Fault Tolerance
SMP	SM	Hom	Dedi	Local	No
COW	DM	Hom	Dedi	Local	No
NOW	DM	Het	Non	Local	Yes
HP Grid	DM	Het	Dedi	Large	No
HT Grid	DM	Het	Non	Large	Yes

Parallel Programming Environments

- Shared-memory:

System	Category	Language binding
PThreads	Operating system	C
Java threads	Programming language	Java
OpenMP	Compiler directives	Fortran, C, C++

- Distributed-memory:

Message passing	RPC or Object-based systems	Grid computing
Sockets	Java RMI	Globus
MPI		Condor

- Hybrid model

Algorithm-Level Parallel Model

- Granularity = Ratio Execution cost between communication / Size Information exchanged
  - Large Granularity → Well suited to large scale architectures
    - Frequency of migration, Size of information exchanged
- Asynchronous exchange more efficient than Synchronous exchange for Heterogeneous or Non-dedicated Architectures
- Fault tolerance for Volatile architectures → Checkpointing (reduced cost)
- Scale : Number of optimization algorithms

Iteration-Level Parallel Model

- Granularity = Evaluation of a partition / Partition communication cost →
  - Adapt granularity to target architecture (size of partitions)
- Asynchronous evaluation is more efficient:
  - Heterogeneous or Non Dedicated or Volatile platforms
  - Heterogeneous computation of the objective function (#solutions #costs)
- Scale: Limited by
  - Size of population
  - Size of Neighborhood.

Solution-Level Parallel Model

- Granularity = Evaluation of sub-function / Cost of solution communication → Not well suited to Large scale and distributed memory architectures
- Limited scalability (number of sub-functions or data partitions)
- Synchronous

Fault-tolerance issues

- Important in volatile, non-dedicated, large-scale platforms
- Checkpointing – Recovery mechanism
  - Application-oriented and NOT System-oriented (reduced complexity in time and space)
  - Algorithmic-level: Current solution or population, iteration number, …
  - Iteration-level: Which partition of population or neighborhood + fitnesses
  - Solution-level: Solution + partial fitnesses

Load Balancing, Security, …

- Static load balancing important for heterogeneous architectures
- Dynamic load balancing important for non dedicated, volatile architectures.
- Security issues important for large-scale architectures (multi-domain administration, firewalls, …) and some applications (medical and bio research, industrial, …)

Performance evaluation

- Speedup ?

$$S_N = T_1 / T_N$$

T_i : Execution time using i processors (wall clock time)
N : number of used processors

  - If S < N ➔ Sublinear speedup
  - If S = N ➔ Linear speedup
  - If S > N ➔ Superlinear speedup

- Absolute speedup
  - Strong: T_1 = Best known sequential algorithm ??
  - Weak :
    - Population of N individuals compared to K islands of N/K individuals
    - Single-solution metaheuristic with N iterations with K S-Meta with N/K iterations

Performance evaluation

- Relative speedup
  - Fixed number of iterations
    - Interesting for evaluating the efficiency of the implementation
    - Superlinear speedup is possible: architecture source (memory, cache, …)
  - Convergence to a solution with a given quality
    - Interesting for evaluation the efficiency of the parallel design (Algorithm-level parallel model)
    - Superlinear speedup possible: search source (such as in branch and bound)
- Heterogenous or Non Dedicated architectures:
  - Efficiency: $E_N = S_N * 100\% / N$ (fraction of time processors are conducting work)
  - 100% efficiency means linear speedup
  - t(j): time for task i, U(i): availability of worker i

$$E = \frac{\sum_{j \in J} t(j)}{\sum_{i \in I} U(i)}$$

- Stochastic : Mean, …

Outline

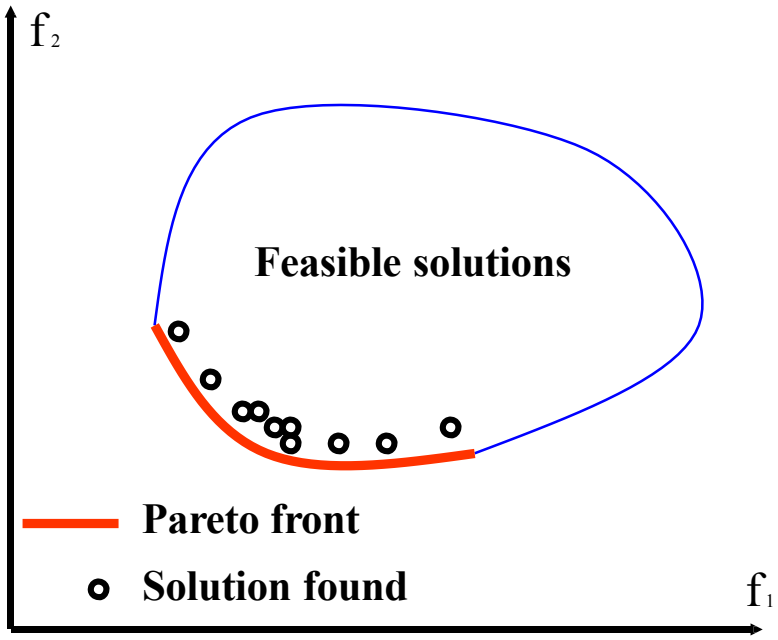
- Parallel Metaheuristics: Design issues
- Parallel Metaheuristics: Implementation issues
  - Hardware platforms, Programming models
- **Adaptation to Multi-objective Optimization**
- Software frameworks
- Illustration : Network design problem

Multi-Objective Optimization

$$\textbf{(MOP)} \quad \begin{cases} \min f(x) = \left(f_1(x), f_2(x), \ldots, f_n(x)\right) & n \geq 2 \\ \text{s.c. } x \in S \end{cases}$$

- Dominance
  - y dominates z if and only if
    $\forall i \in [1, \ldots, n], y_i \leq z_i$
    and $\exists i \in [1, \ldots, n], y_i < z_i$
- Pareto solution
  - A solution x is Pareto if a solution which dominates x does not exist

➔ Goal: Find a good quality and well diversified set of Pareto solutions



Algorithm-Level Parallel Model

- General and Global Cooperative Models:
  - Which Information to exchange ?
    - Pareto archive or current populations (random, elite, uniform, …), …
  - Any replacement strategy for the current population
    - Based on: Dominance, Indicator, Scalarization, …
- Partial:
  - Decompose the Pareto front (objective space)
  - # dominance criteria
- Specialist :
  - Solve # problems (# objectives subsets)

Iteration-Level Parallel Model

- Many MOP applications with complex objectives (CFD – computational fluid dynamics, CEM – computational electromagnetics, FEM - finite element method, …)
- Fitness assignment
  - Dominance, performance indicator, … : complex procedures to parallelize
- Elitism
  - Pareto archiving : complex procedure to parallelize

Solution-Level Parallel Model

- Decomposition of the *n* objectives
- Multi-disciplinary Design Optimization
  - Many engineering domains with different models (# disciplines, #solvers)
  - Ex: Car design
  - Optimize the air flow around a car → computational fluid dynamics (CFD) solver
  - Optimize the toughness of materials → finite element method (FEM) solver

Outline

- Parallel Metaheuristics: Design issues
- Parallel Metaheuristics: Implementation
  - issues
    - Hardware platforms, Programming models
- Adaptation to Multi-objective Optimization
- **Software Frameworks for Parallel Metaheuristics**
- Illustration : Network design problem

Why ?

- From scratch: high development cost, error prone, difficult to maintain, …
- Code reuse: difficult to reuse, adaptation cost, …
- Design and code reuse – software components: Hollywood principle « Don't call us, we call you »

Combinatorial Optimization Problems (COPs) in practice:

- Diversity
- Continual evolution of the modeling (regards needs, objectives, constraints, …)
- Need to experiment many solving methods, techniques of parallelization, hybridization, parameters, …

Design Objectives

- Maximal Reuse of code and design
  - Separation between resolution methods and target problems
    - Invariant part given
    - Problem specific part specified but to implement
- Flexibility et Adaptability
  - Adding and updating other optimization methods, search mechanisms, operators, encoding, ...
  - … to solve new problems
- Utility
  - Large panel of methods, hybrids, parallel strategies, …
- Portability
  - Deployment on different platforms (Standard library)
- Transparent access to performance and robustness
  - Parallel implementation is tranparent to the target hardware platform
- **Open source, Efficiency, Easy to use, …**

Examples

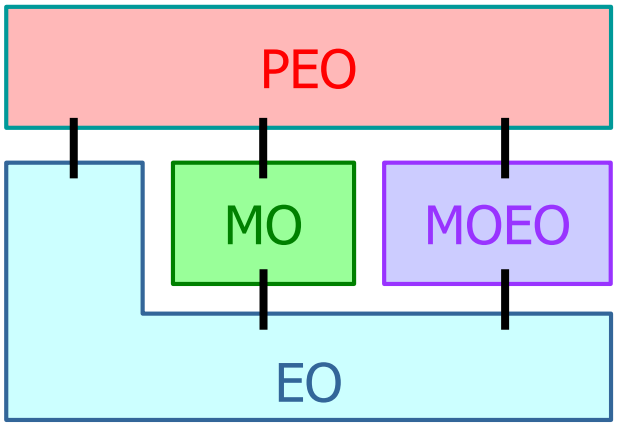
	Metaheuristics	Parallelism support at design	Parall. and dist. at implementation
ECJ	E.A.	Island cooperation	Threads / Sockets
D. BEAGLE	E.A.	Centralized model / Island cooperation.	Sockets
J-DEAL	E.A.	Centralized model	Sockets
DREAM	E.A.	Island cooperation	Sockets / P2P
MALLBA	L.S. / E.A.	All	MPI, Netstream
PARADISEO	S-Meta / P-Meta	All	MPI, Condor, PThreads, Globus, CUDA

PARADISEO (PARAllel and DIStributed Evolving Objects)

- PARADISEO in some words …
  http://paradiseo.gforge.inria.fr

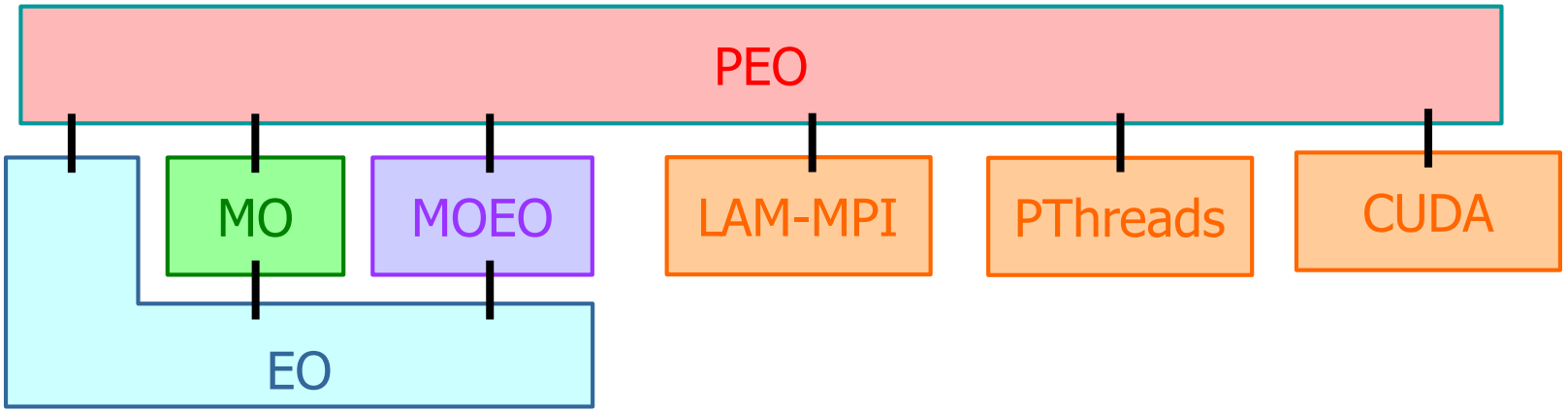
- An Open Source C++ framework (STL-Template)
- Paradigm-free, unifying metaheuristics
- Flexible regards the tackled problem
- Generic and reusable components (operators of variation, selection, replacement, criterion of termination, …)
- Many services (visualization, management of command line parameters, check-pointing, …)

PARADISEO : http://paradiseo.gforge.inria.fr



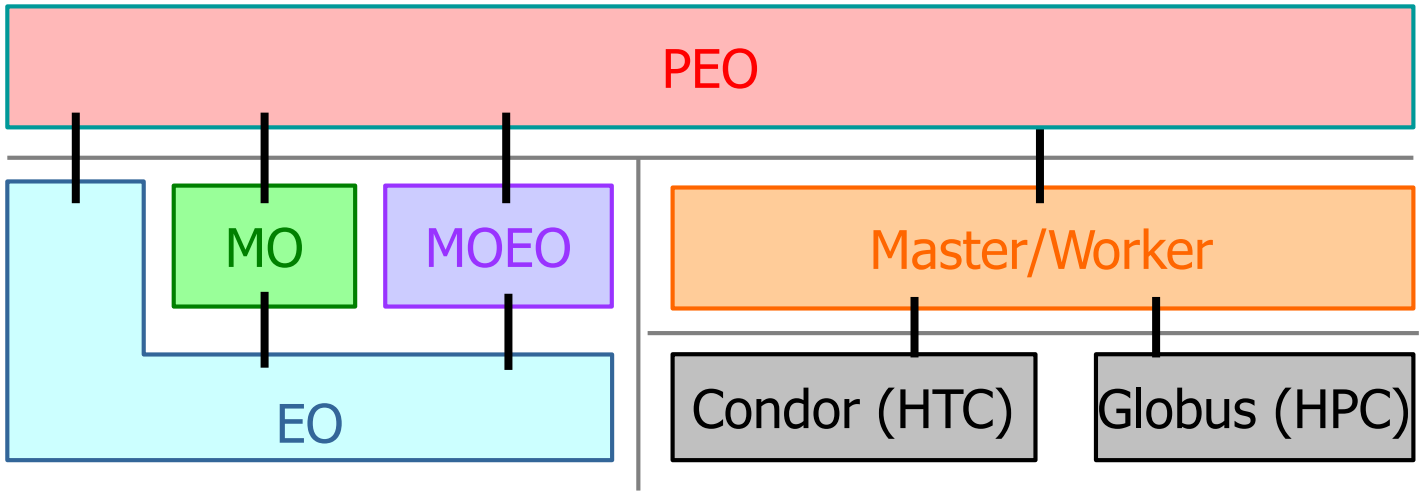
- Evolving Objects (EO) for the design of population-based metaheuristics: GA, GP, ES, EDA, PSO, …
- Moving Objects (MO) for the design of solution-based metaheuristics: LS, TS, SA, VNS, ILS
- Multi-Objective EO (MOEO) embedding features and techniques related to multi-objective optimization,
- PEO for the parallelization and hybridization of metaheuristics

Architecture (level of execution)
Parallel and distributed platforms



- Parallelism and distribution
  - Communication libraries (MPI LAM)
    → Deployment on networks/clusters of stations (COWs, NOWs)
  - Multi-threading layer (Posix threads)
    → multi-core, multi-processors with shared memory (SMPs)
  - CUDA environments
    → GPUs
  - Transparent to the user

Architecture (level of execution)
Grid computing



- Gridification
  - Re-visit parallel models taken into account the characterisitics of Grids
  - Coupling of ParadisEO with a Grid middleware (Condor-MW and Globus)

- Transparent volatility & checkpointing
  - Ex : Definition in ParadisEO-CMW of the memory of each metaheuristic and the associated parallel models

Illustration: Core classes of the Local Search

UML notation
(Unified Modeling Language)

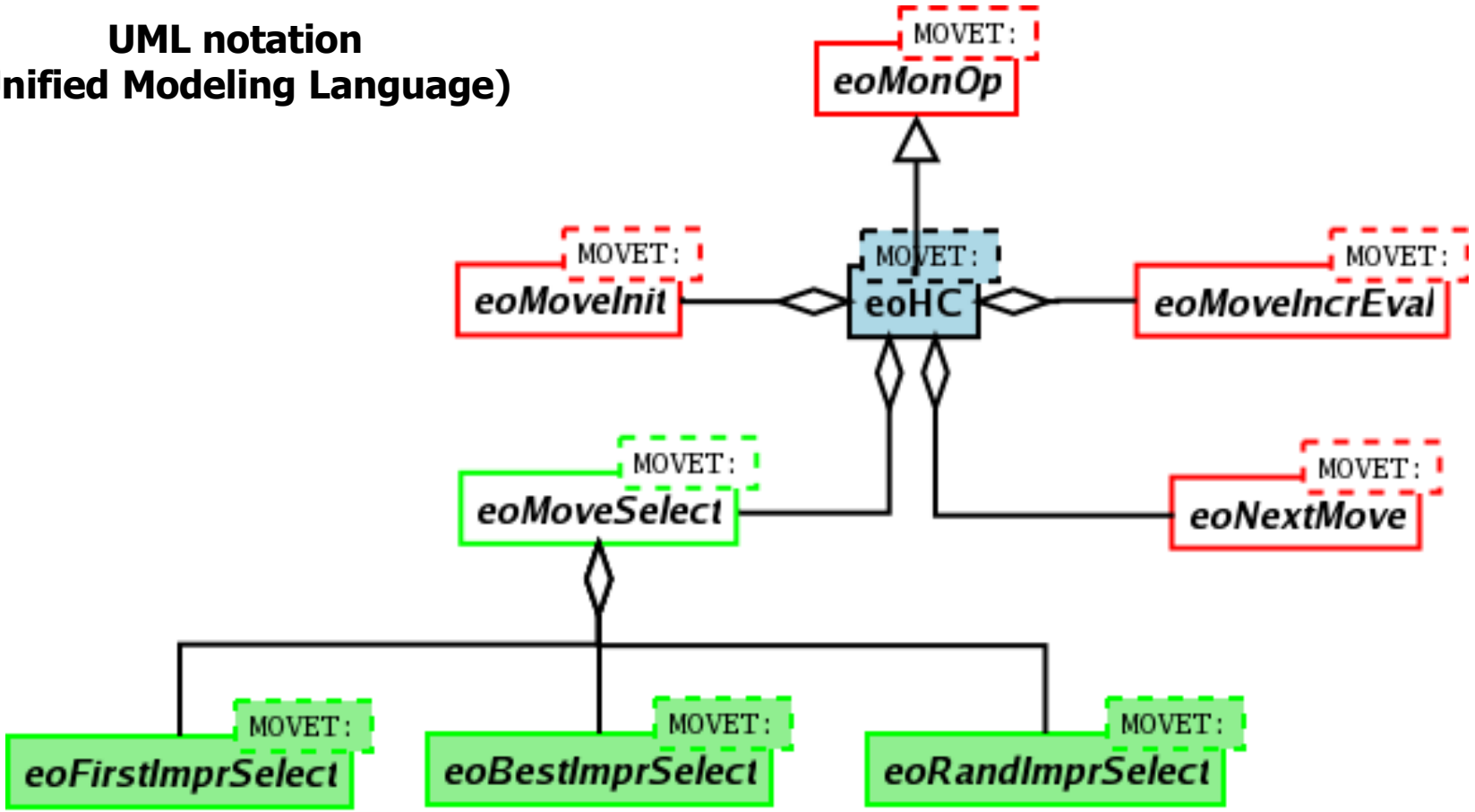


Illustration: Core classes of the Evolutionary Algorithm

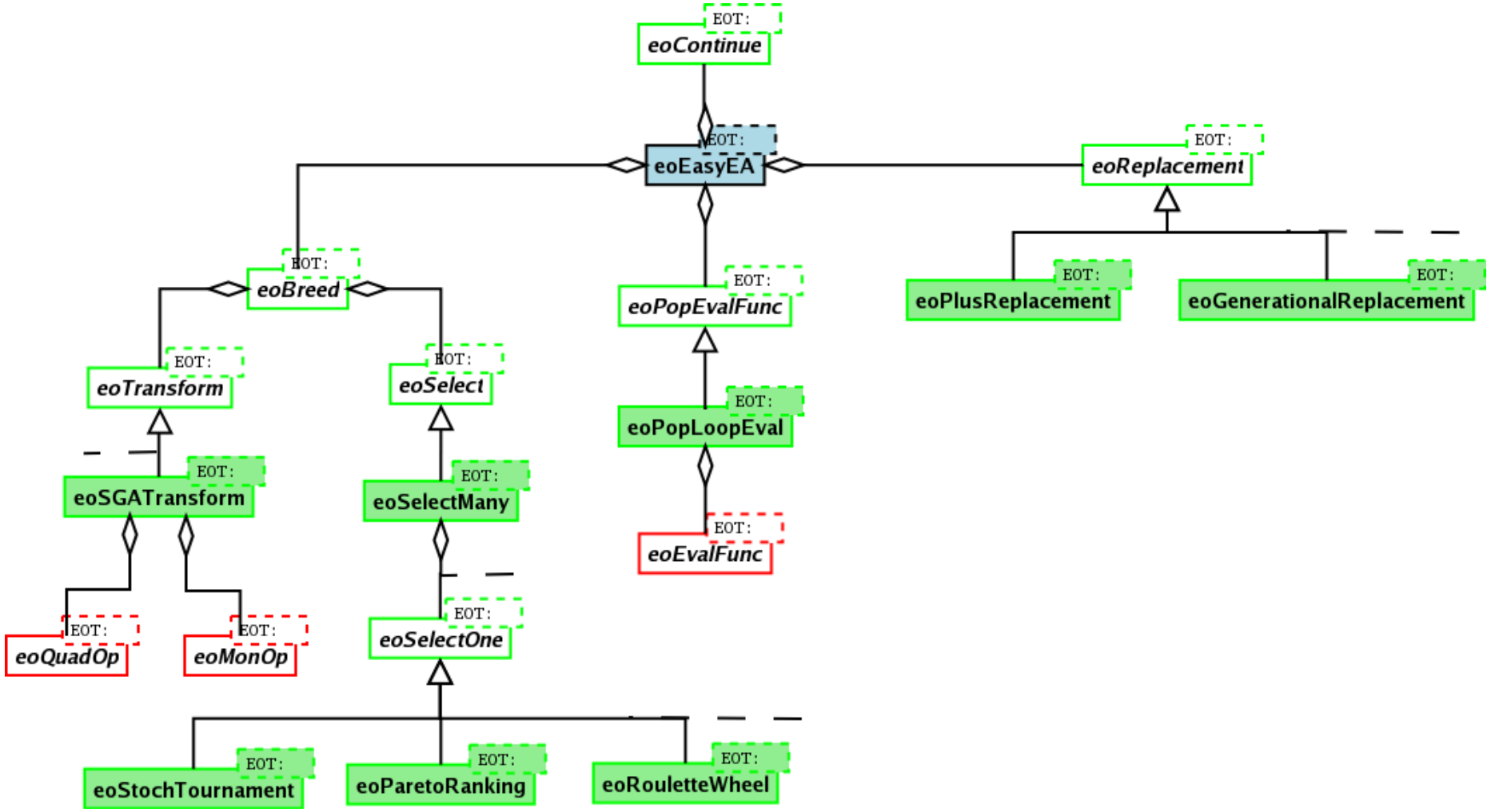


Illustration: The cooperative island model of

```
/* To enable migrations (i.e. exchanges
   of individuals) */
eoPopChan <Route> pop_chan ;
/* Migrations will occur periodically */
eoFreqContinue <Route> mig_cont (FREQ_MIG) ;
/* Migrations are composed of random individuals
*/
eoRandomSelect <Route> mig_select_one ;
/* Selector of NUM_EMIG emigrants*/
eoSelectNumber <Route> mig_select
                (mig_select_one,    NUM_EMIG) ;
/* Emigrants replace the worst individuals*/
eoPlusReplacement <Route> mig_replace ;
/* The ring topology */
eoRingTopology topo (naming_chan) ;
/* Building a manager of migrations */
eoDistAsyncIslandMig <Route> island_mig
                    (naming_chan
                    , pop_chan,
                    mig_cont,
                    mig_select,
                    mig_replace,
                    pop,
                    pop,
                    topo)
                    ;
```

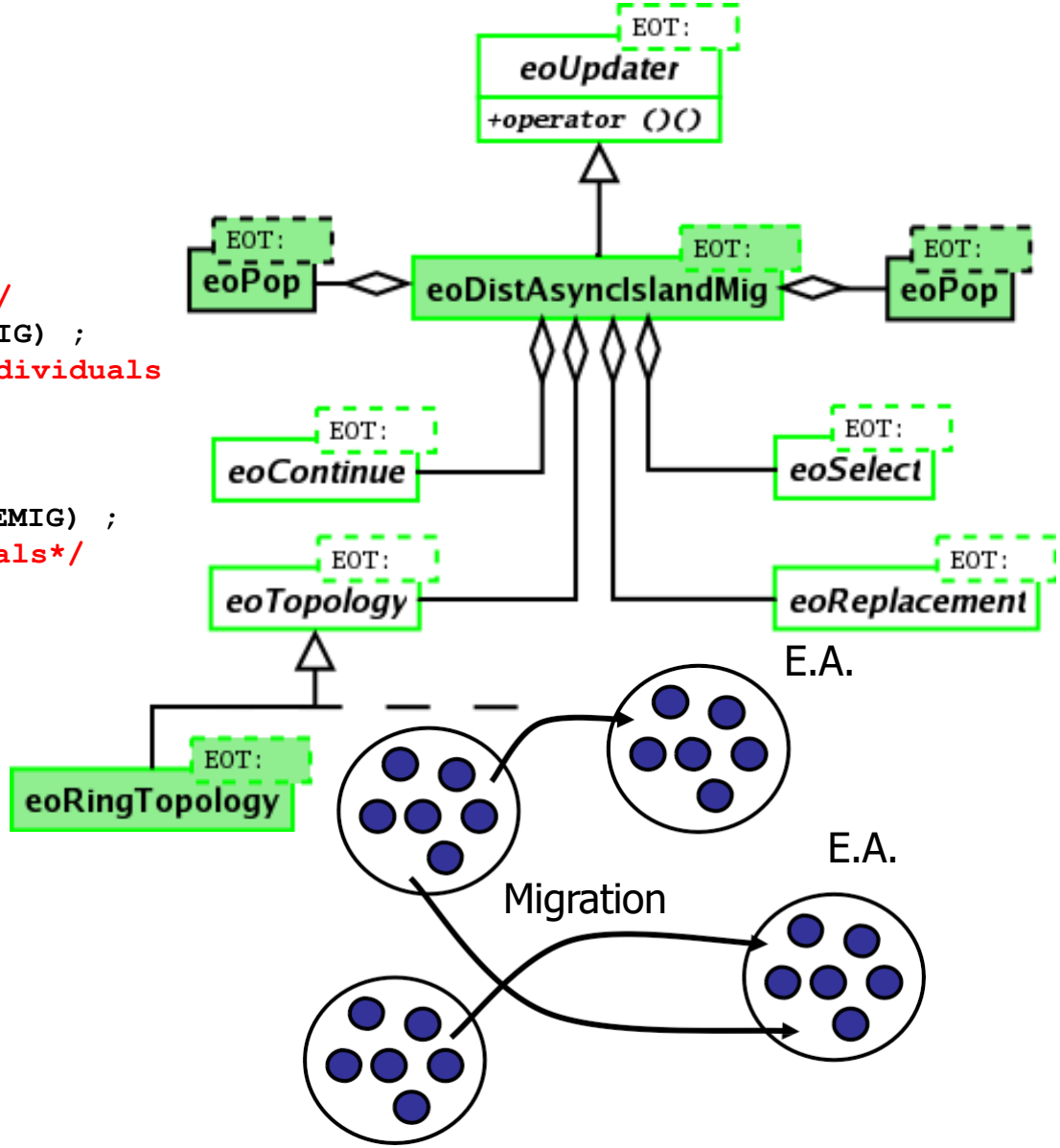
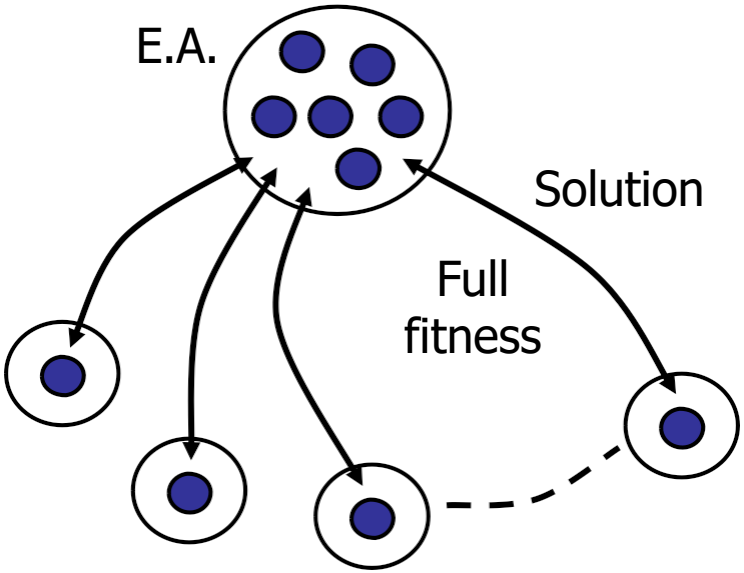


Illustration: The parallelization of the evaluation step



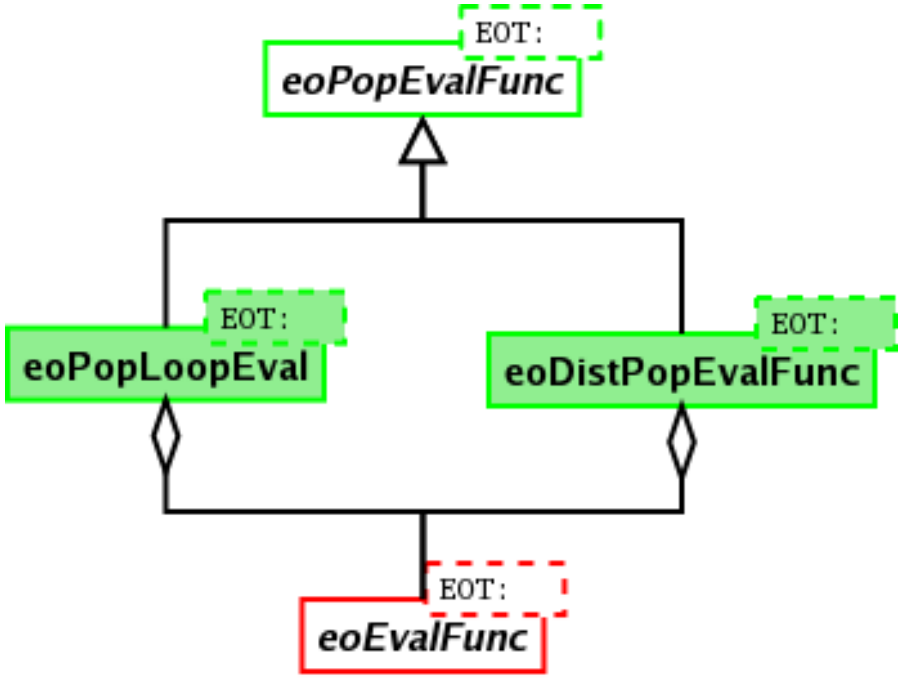
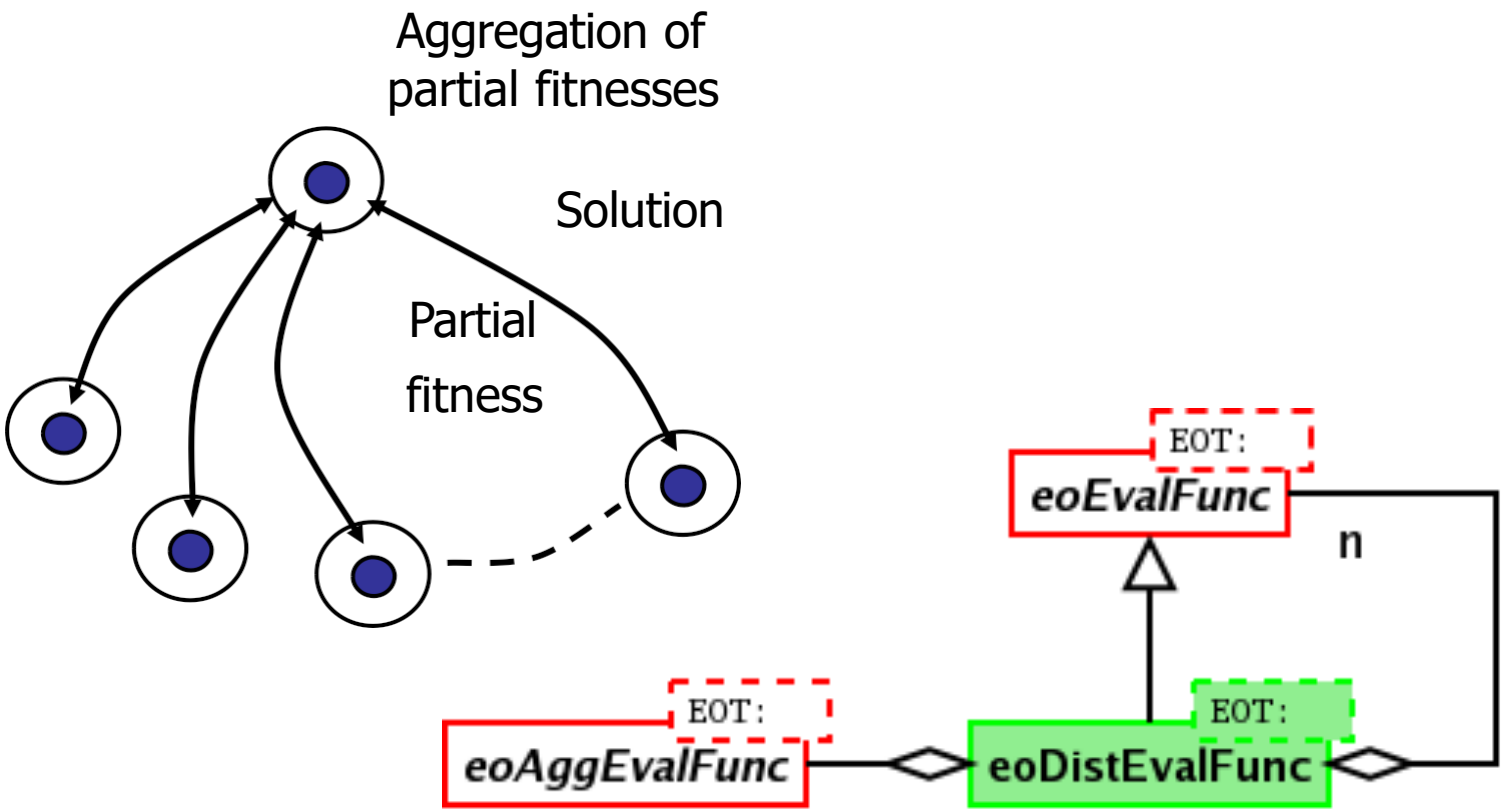


Illustration: The parallelization of the objective function



Outline

- Parallel Metaheuristics: Design issues
- Parallel Metaheuristics: Implementation issues
  - Hardware platforms, Programming models
- Adaptation to Multi-objective Optimization
- Software frameworks
- **Illustration : Network design problem**
  - **Combined use of the 3 parallel models**
  - **Multi-objective problem**
  - **Implemented on COWs, NOWs, HPC Grids and HTC Grids**
  - **Using of PARADISEO – EO & MO & MOEO & PEO**

Design of radio networks in mobile telecommunication

- Network design
  - Positioning sites
  - Fixing a set of parameters for each antenna
- Multi-objective problem
  - Cost of the network: Number of sites
  - Quality of Service
- NP-hard problem with:
  - Huge search space
  - High cost (CPU and memory) objective functions, constraints.

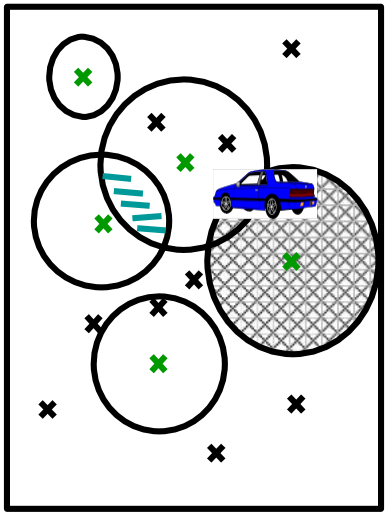
A brief description

- A set of base stations that satisfy the following constraints …
  - Cover
  - Handover
- … and optimizes the following criterion
  - Min. the number of sites
  - Min. interferences
  - Max. yield traffic

Parameters of antennas

Données	bornes
Ps	[26, 55] dBm
Diagram	3 types
Hauteur	[30, 50] m
Azimut	[0, 359] °
Inclinaison	[-15, 0] °
TRX	[1, 7]

Working area



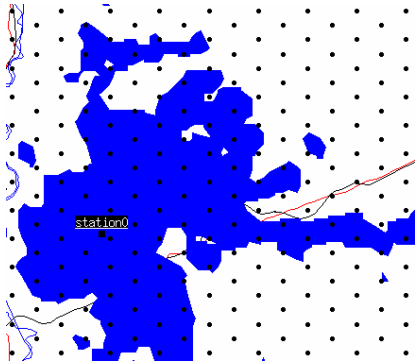
Used sites

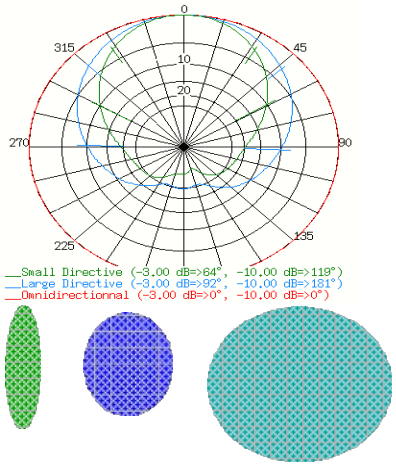
Useless sites

Handover area

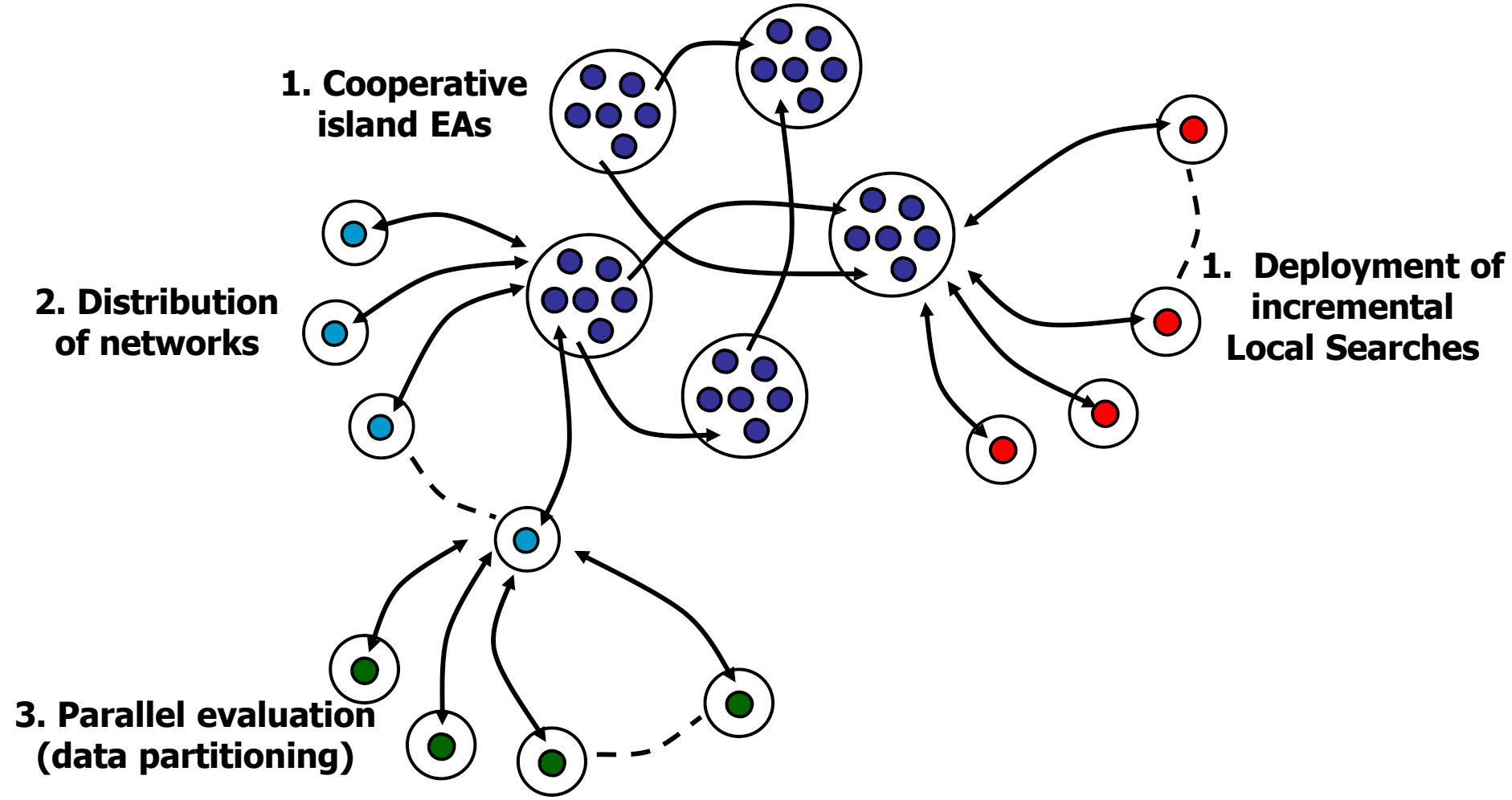
Cover area

Propagation model (Free spaceOkumura-Hata)



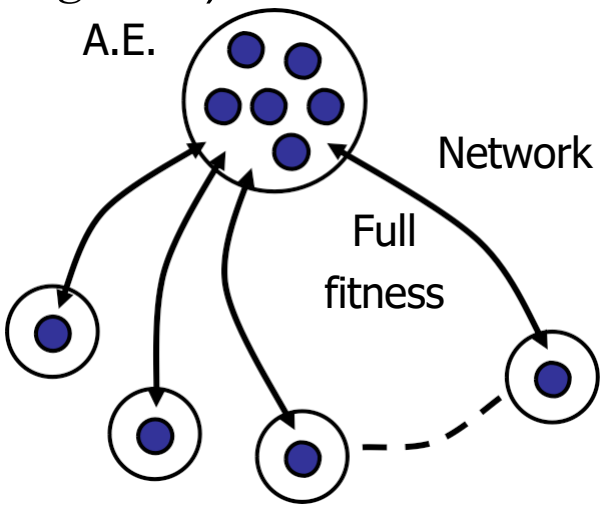


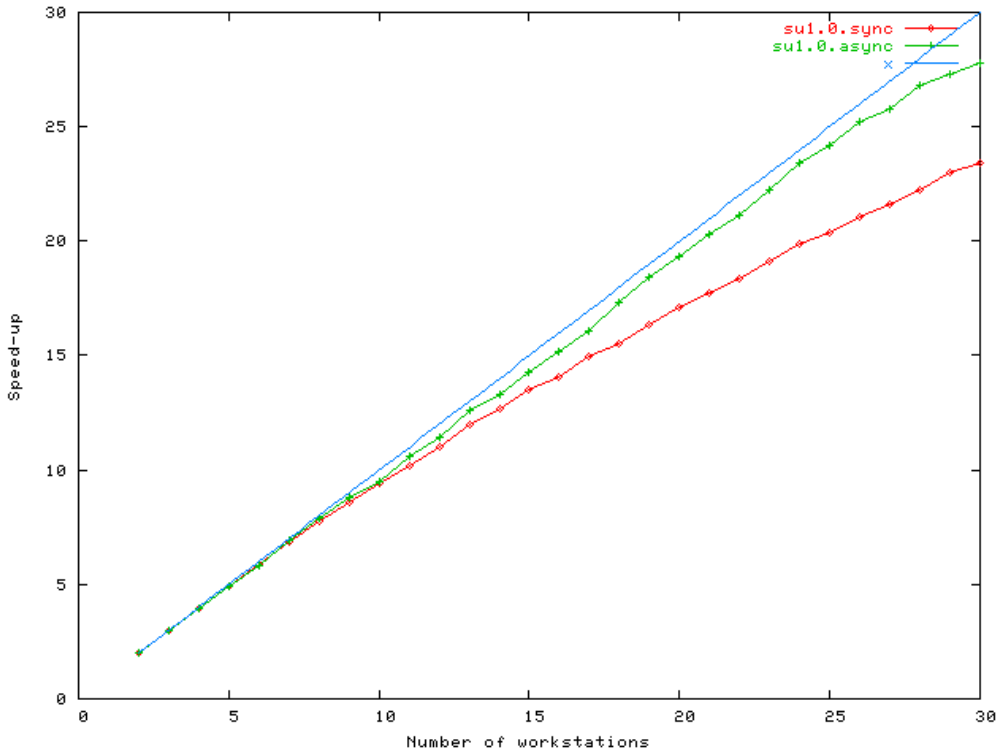
A multi-layer hierarchical parallel/hybrid metaheuristic



Iteration-level Parallel Model (Homogeneous and Dedicated Cluster)

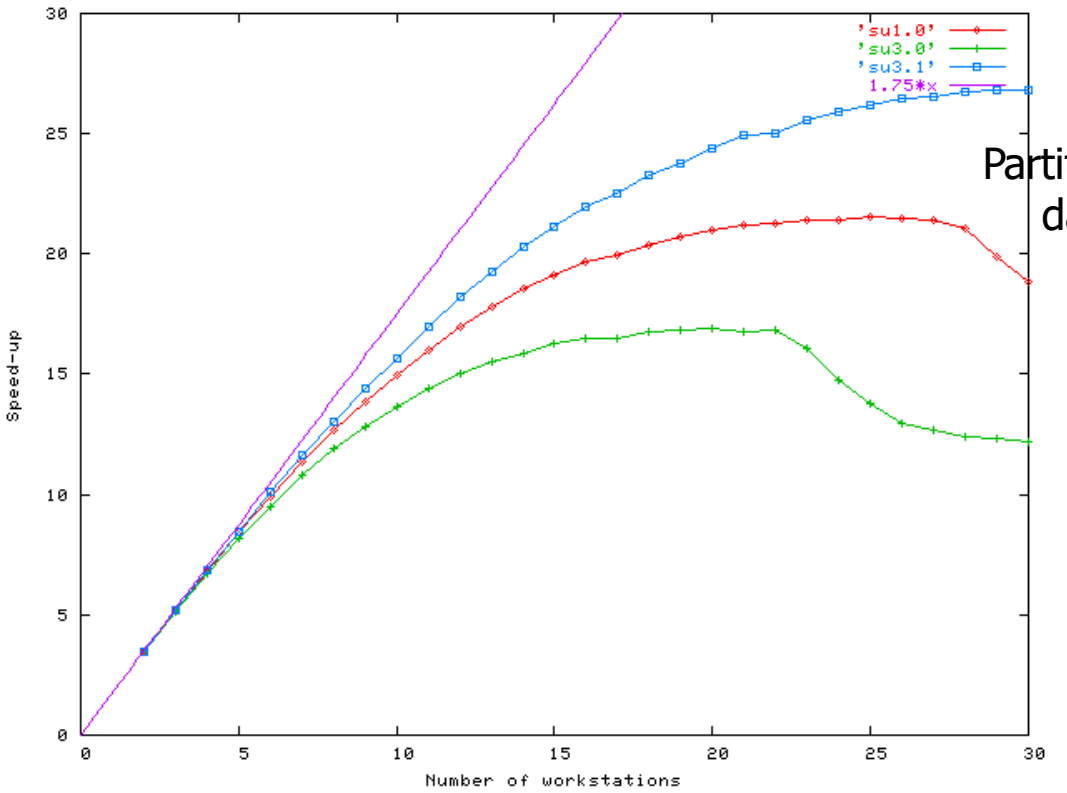
- Synchronous/Asynchronous
- Deploying irregular tasks
  → The computation time is dependent of the number of activated sites of the network
- Limited scalability of the synchronous model (size of the pop., *e.g.* 100)

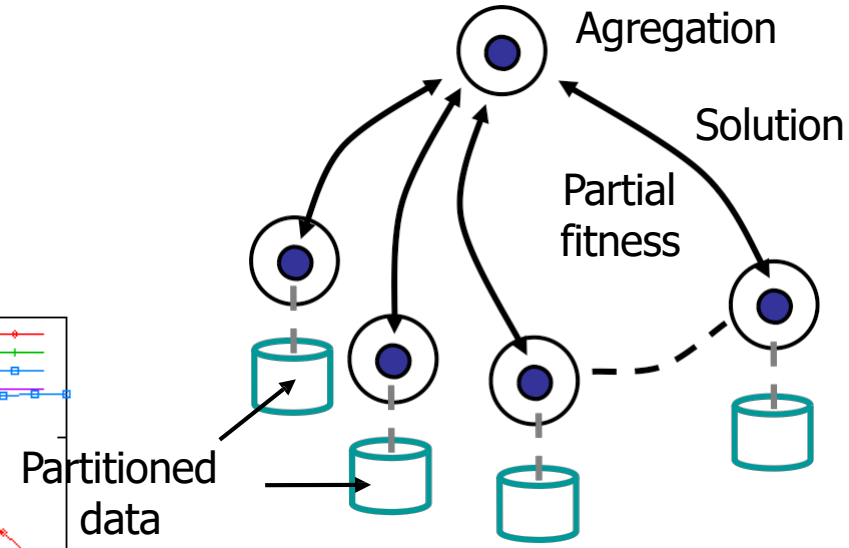


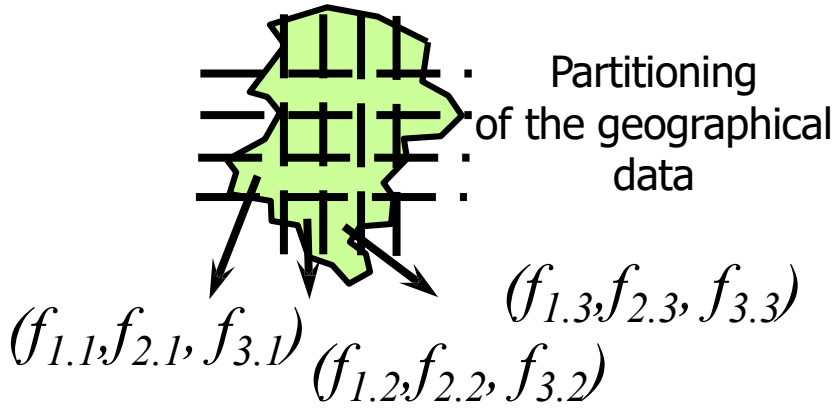


Solution-level Parallel Model

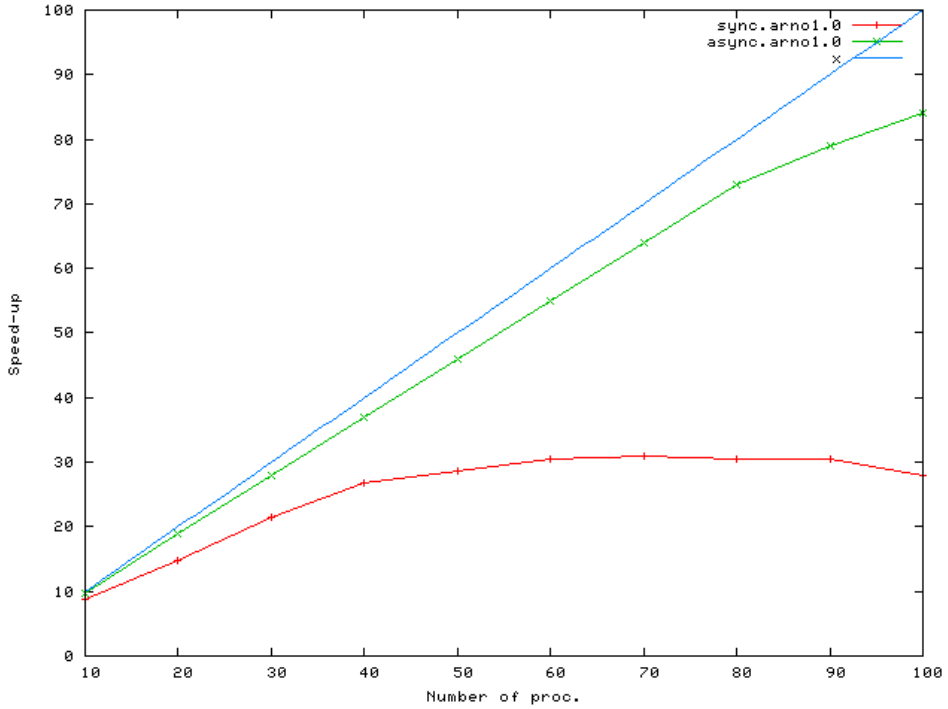
- Synchronous, fine-grained !
- Larger instances
- Poor scalability

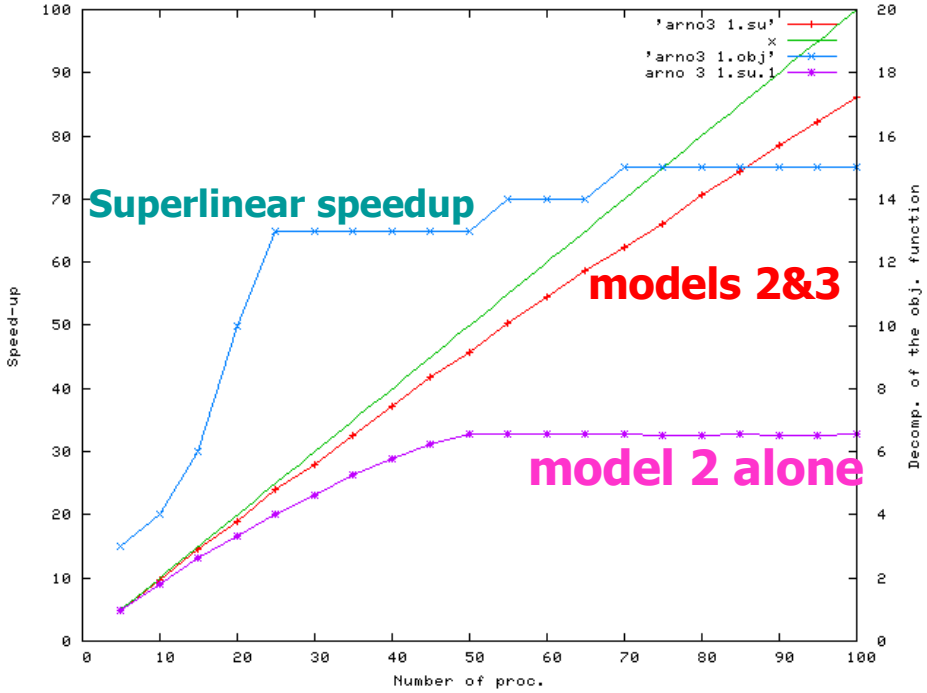






Experimentation under PARADISEO (non-dedicated cluster of PCs)





Parallel evaluation of the population (model 2)

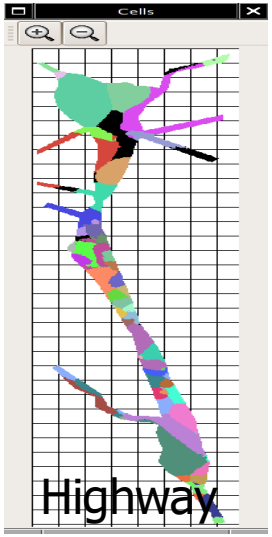
Synchronous vs. Asynchronous

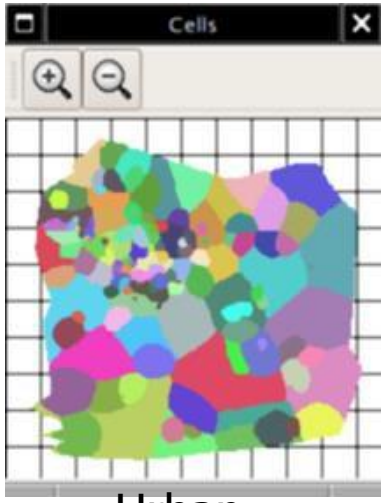
Parallel Evaluation of a solution (model 3)

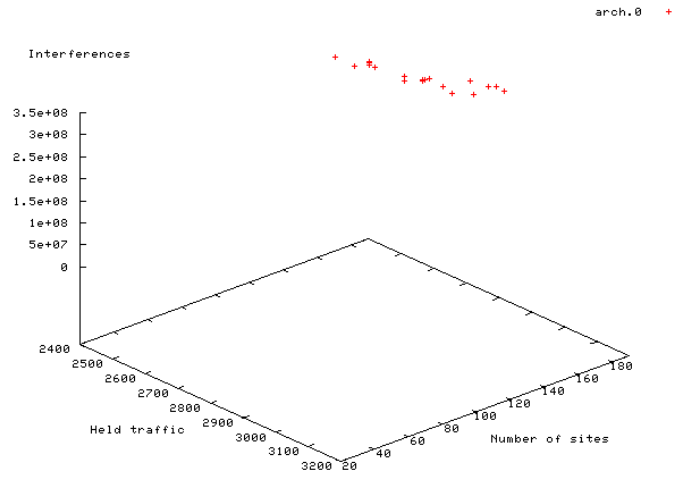
Influence of the granularity on the efficiency (synchronous)

High-Throughput Computing Grid: Campus of Lille (3 # administrative domains)

Platform	HTC Grid (Polytech, IUT, LIFL)
Prog. Environment	**Condor**
Number of proc.	100 (heterog. and non dedicated)
Cumulative wall clock time	30681 h.
Wall clock time	Almost 15 days
Parallel efficiency	**0.98**



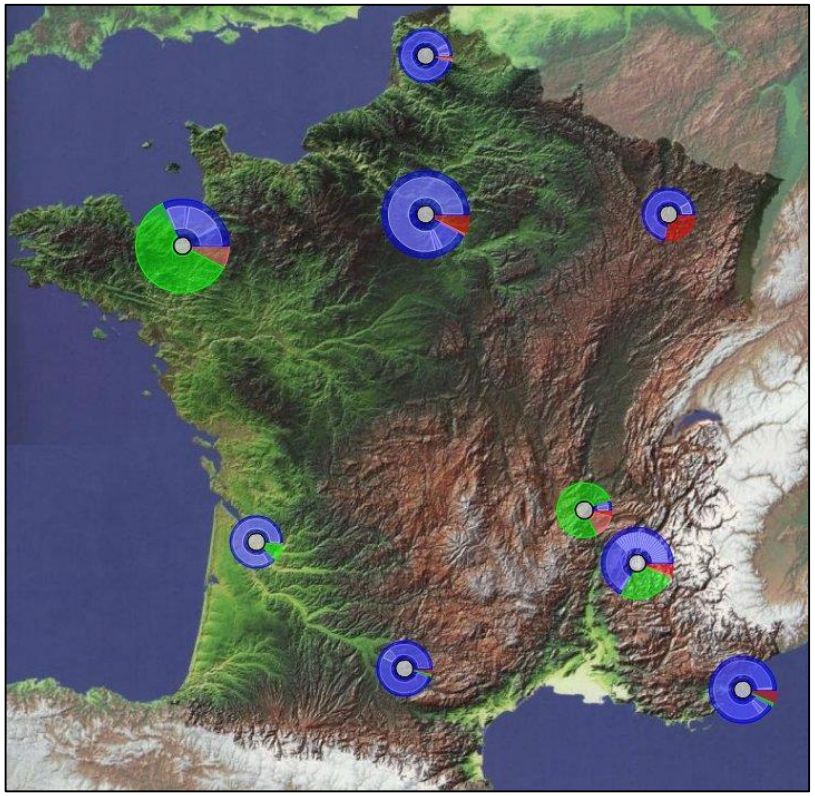




High-Performance Computing Grid: GRID'5000 under Globus

- 400 CPUs on 6 sites: Lille, Nice-Sophia Antipolis, Lyon, Nancy, Rennes

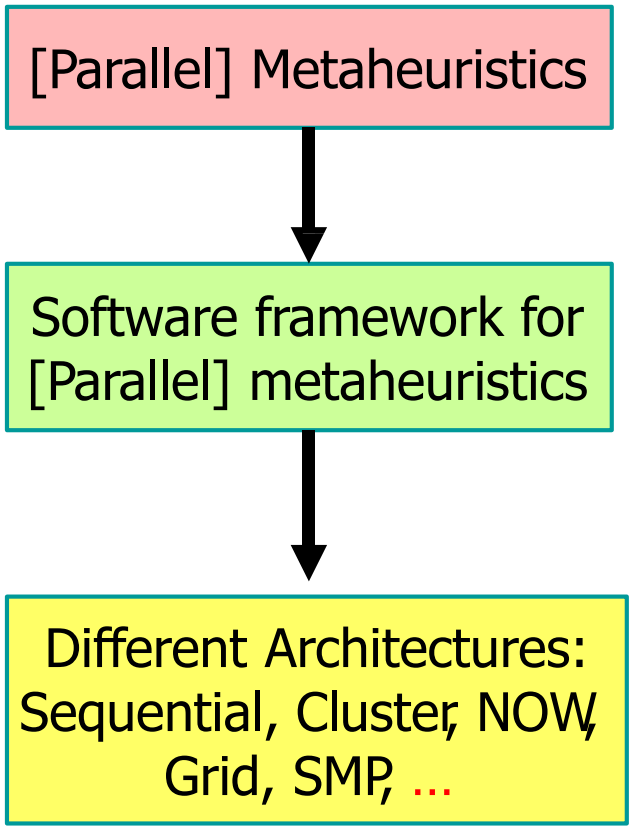
- **Parallel efficiency = 0.92**
- **Best results obtained**
- **More than 22 years of cumulative wall clock time (other benchmark on 2465 processors)**



GRID'5000: A fully reconfigurable grid! : Linux « images » having **Globus** and MPICH-G2 already installed.

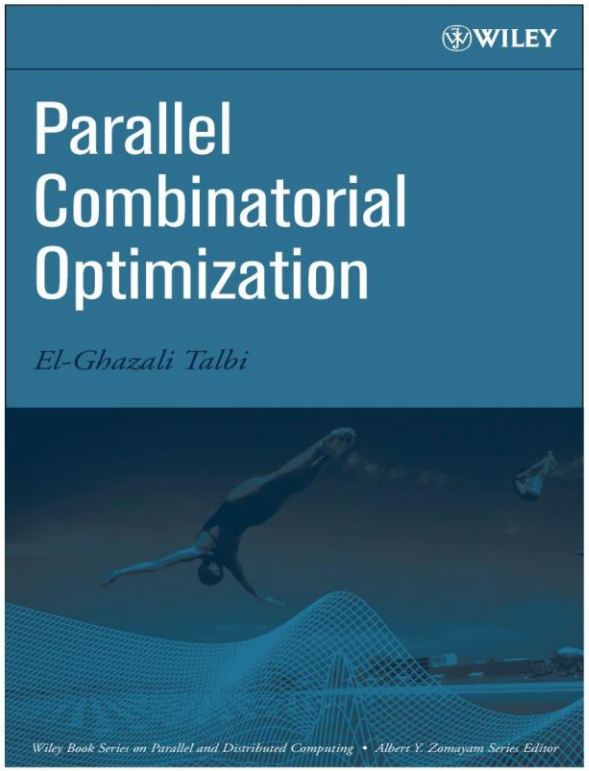
Conclusions

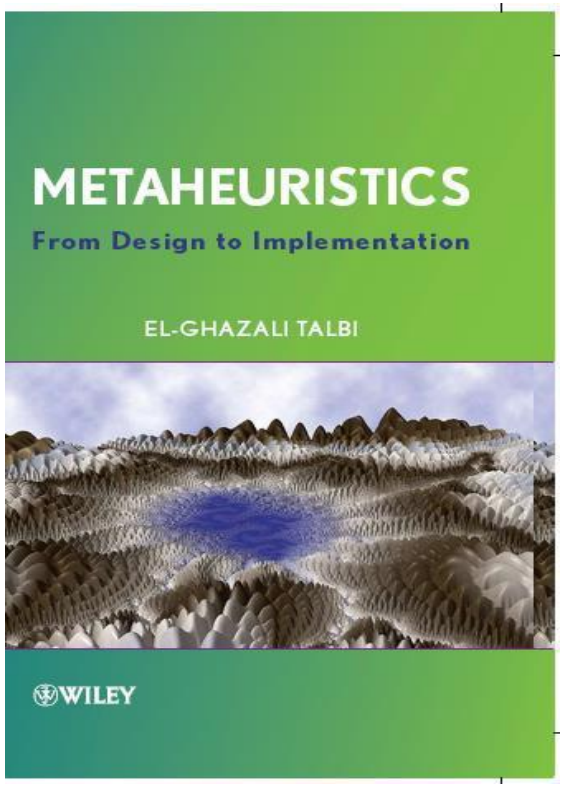
- Unifying Parallel Models for Metaheuristics
- Clear separation between parallel design and parallel implementation
- Encourage the use of software framework for [parallel] metaheuristics



Perspectives

- Parallel models combining Metaheuristics & Exact methods (Algorithms, Coupling of Software, …)
- Parallel models for dynamic and robust optimization problems
- Parallel models for optimization problems with uncertainty
  - Need a multiple evaluation of a solution
- Solving challenging problems on Grids (Ex: Molecular biology, Engineering design, …)
- Metaheuristics on heterogeneous architectures (GPU+multi-cores)


WILEY
Parallel Combinatorial Optimization
El-Ghazali Talbi
Wiley Book Series on Parallel and Distributed Computing • Albert Y. Zomaya Series Editor


METAHEURISTICS
From Design to Implementation
EL-GHAZALI TALBI
WILEY