EGEE'09: Bio-inspired Algorithms in Grid

DIOGENES: Application Oriented Task Scheduling Using Genetic Algorithms

Presenter: Florin Pop
University Politehnica of Bucharest

[1]Gabriel Neagu, [2]Florin Pop, [2]Valentin Cristea.

[1]National Research Institute in Informatic (ICI)
[2]University "Politehnica" of Bucharest

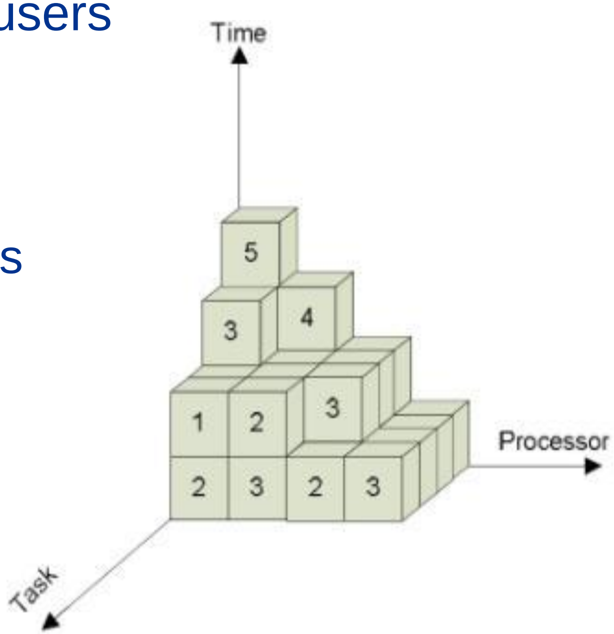
e-infrastructure

The Importance Grid Scheduling

- **The scheduling problem**
  - Allocate a group of different tasks to available resources
  - NP-Complete problem
  - Optimizations of approximate algorithms are required
  - Dynamic (available) resource allocation

- **Environment: Computational and data GRID**
  - *Heterogeneous* – resources with different processing capacities
  - *Shared* – resources can be used by multiple users

- **Objectives**
  - Optimize the application execution
    - Minimize the total execution time of the tasks
    - Efficient use of computing resources
  - Successful completion of tasks
    - Deadline restrictions
    - Resource requirements
  - Optimize the scheduling process



Agenda

- **Scheduling Requirements and Objectives**
- **Scheduling Model (DIOGENES)**
  - System Anatomy
  - Functionality Aspects
- **Genetic Algorithm used in DIOGENES**
  - Chromosome Encoding
  - Genetic Operators
  - Fitness Function
- **Experimental Results**
  - Algorithm Convergence
  - Decentralization
  - Estimate Times Versus Real Execution Times
  - Comparison of Various Scheduling Methods
- **Conclusions**

Scheduling Requirements

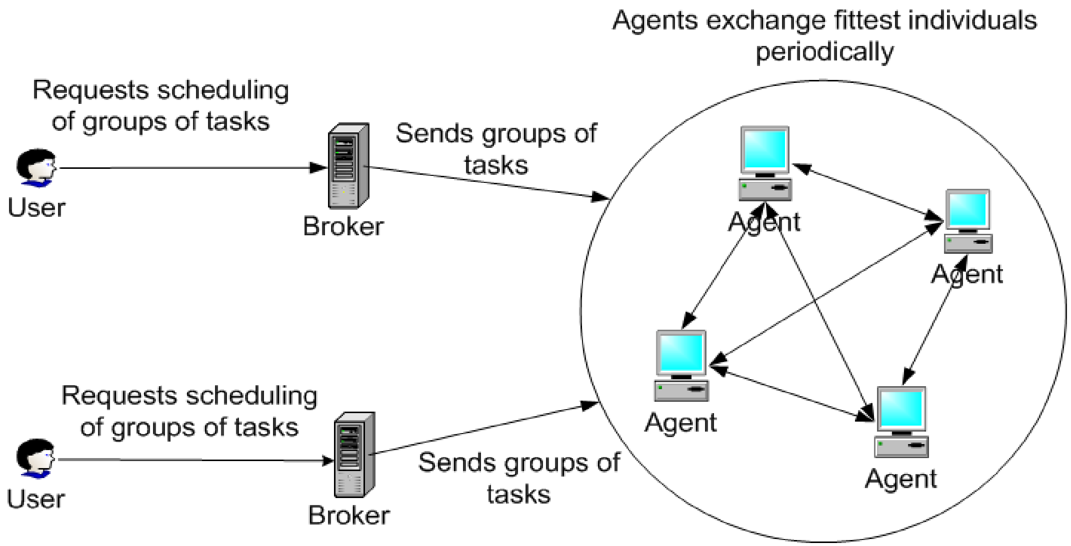
- **Real-time task scheduling**
  - comply with deadlines
  - use monitoring information in the process of scheduling
- **General assumptions**
  - Heterogeneous environment
    - *Tasks* – different execution times, memory required, deadlines, *etc*.
    - *Resources* – different CPU power, memory, swap space, *etc.*
  - *Meta-task*: a group of multiple tasks.
    - Example: A group of image processing applications operating on different images
  - No multitasking
    - Each node executes a single task (in a queue) at a time (Condor, PBS work in this way).

Objectives

- **Makespan**
  - Time from when first operation starts to last operation finishes.
- **Flow times**
  - Time when a job is ready to when the job finishes.
- **Deadlines**
  - Lateness
  - Tardiness
  - Unit penalty
- **Priority**
  - If each job has varying importance, prioritied can also be used.

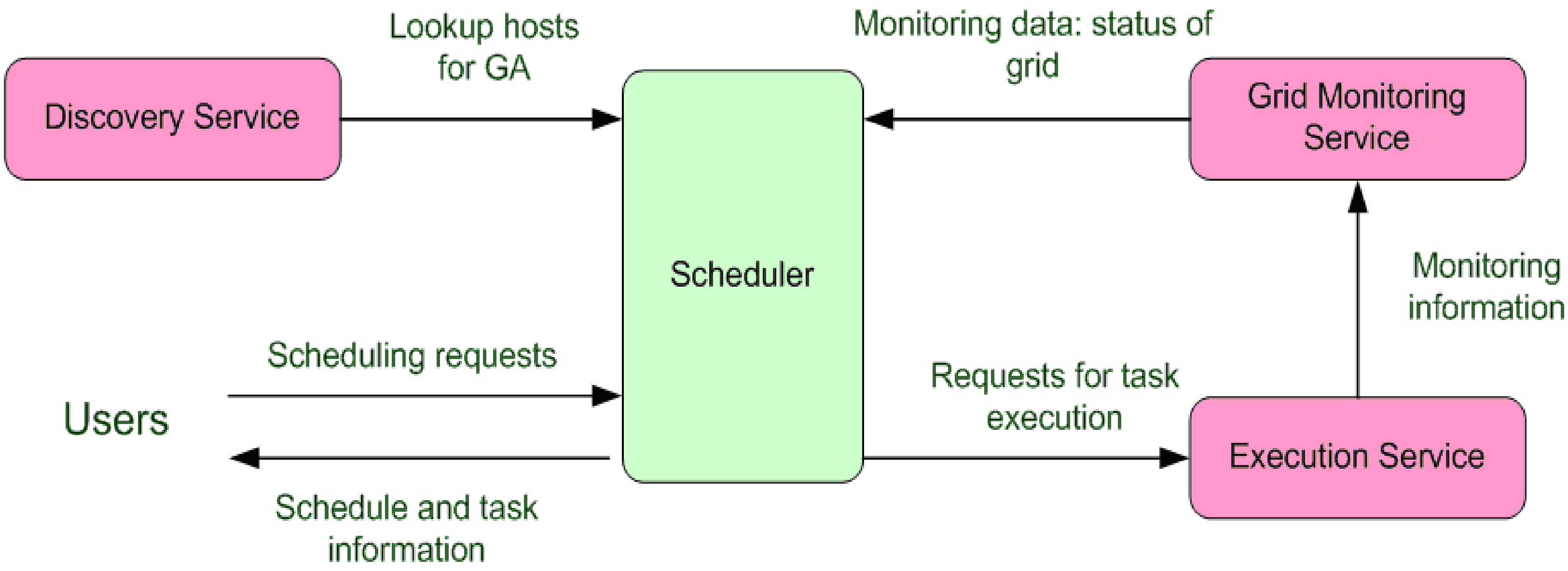
Scheduling Model

- **Broker**
  - Collect user requests: the groups of tasks to be scheduled (input of the algorithm)
  - Can be remote or on the same workstation with an Agent
- **Agent**
  - Run the scheduling algorithm using monitoring information and the tasks from Broker
  - Migration of the best current solution
  - Communication strategy: *synchronous* (similar computing resources) and *asynchronous* (different computing power on resources)



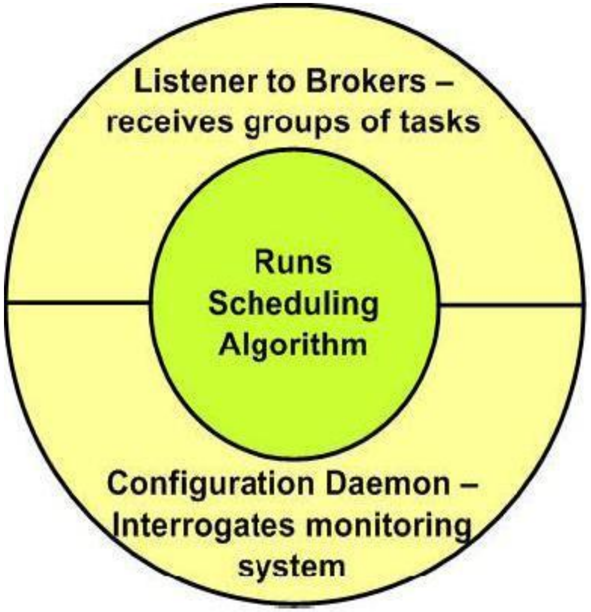
System Anatomy

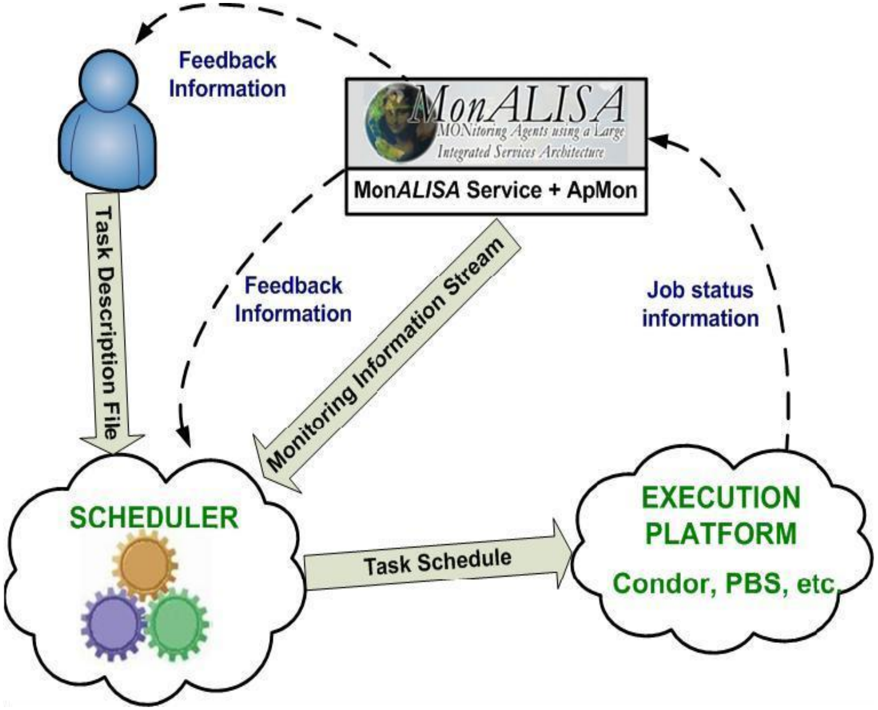
- Users submit requests for task allocations
- Simultaneous requests from various workstations can be submitted
- Obtain real-time information about the state of the Grid
- Incorporate the monitoring information in the algorithm used for scheduling



Functionality Aspects

- **Agent's Anatomy: two-layered structure:**
  - Core : runs the scheduling algorithm
  - Shell : provides information for the Core
    - the Listener to Brokers – receives groups of tasks
    - the Configuration Daemon – obtains up-to-date information about the state of execution resources
- **Input {Task Description File, Monitoring Information Stream}**
- **Output {Task Schedule}**

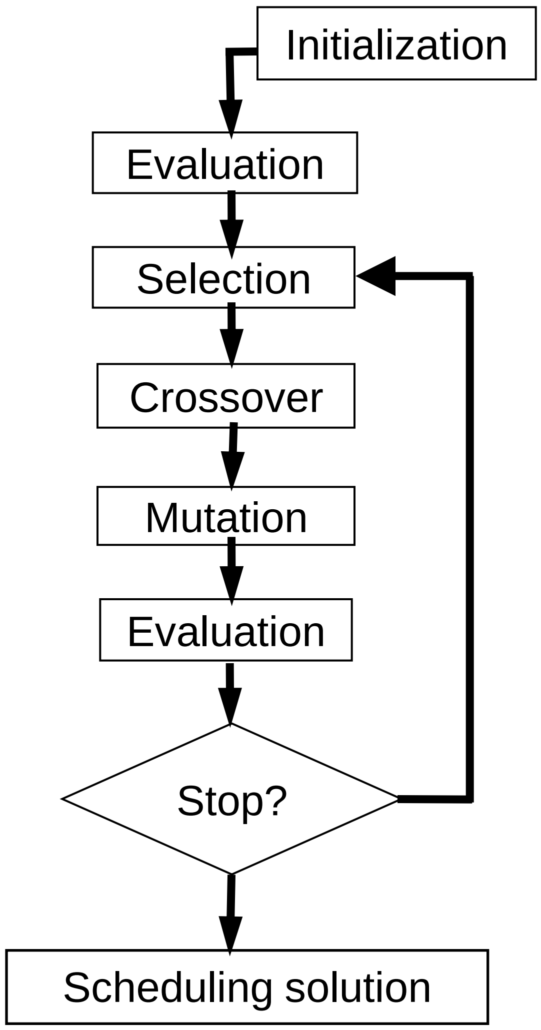




Genetic Algorithm

- **Are usually applied in optimization of NP-Complete problems**
- **Offers a near-optimal solution for scheduling problem**

Representation of a chromosome	Vector of genes
Initialization	Random mapping of tasks on the existing computational resources
Evaluation	Fitness function based
Parents selection	Roulette-wheel selection
Selection results	The offspring replace the parents in the new population
Recombination	Single-point crossover
Mutation	Assignment of a task to another processor with adaptive probability
Specialization	The crossover operation has the highest probability



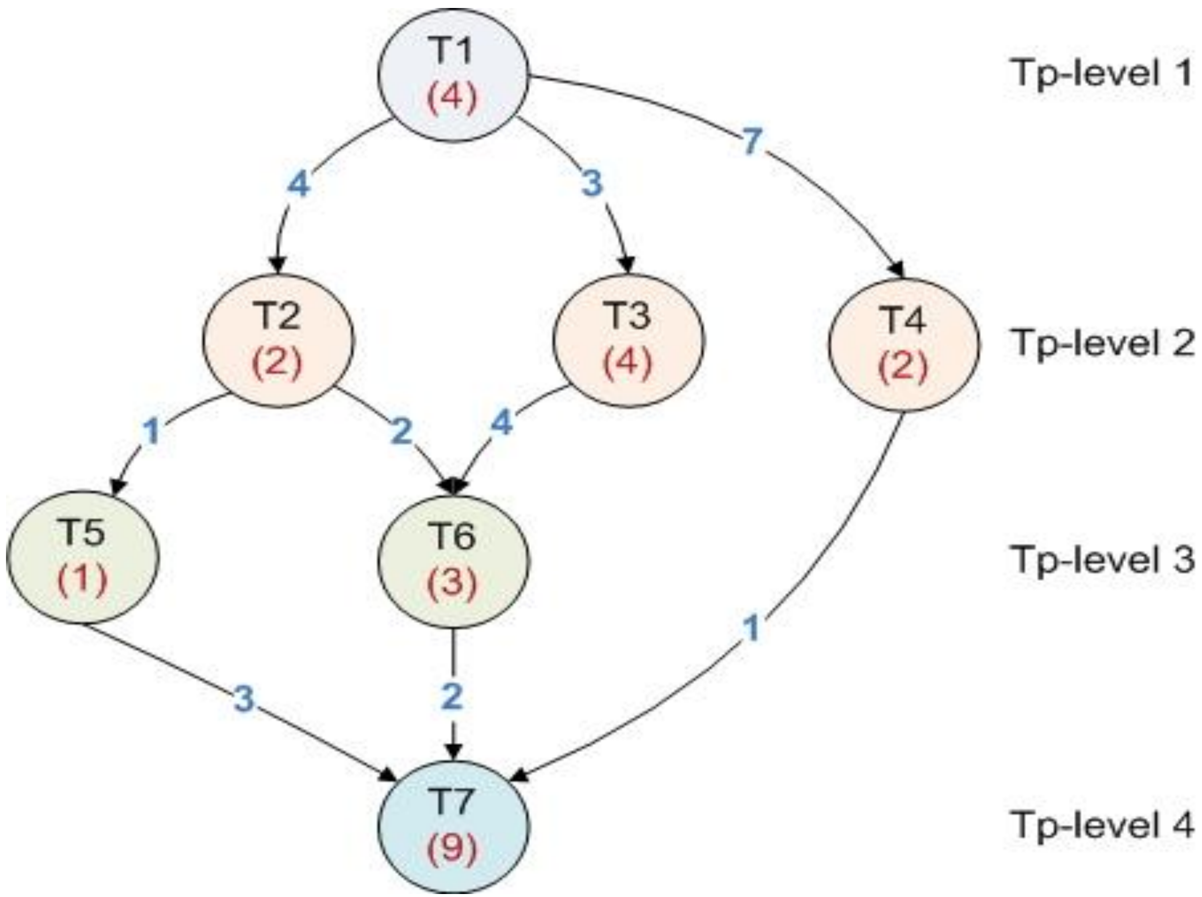
Chromosome Encoding

- **Gene representation**
  - tuple [Task, Processor]
- **Chromosome representation (no dependencies)**
  - vector of genes having a fixed length
  - illustrates a possible mapping of the tasks on the existing processors

1	2	3	4	5	6	7	8
[T2,P0]	[T5,P2]	[T0, P2]	[T1, P1]	[T3, P0]	[T4, P1]	[T7, P3]	[T6, P1]

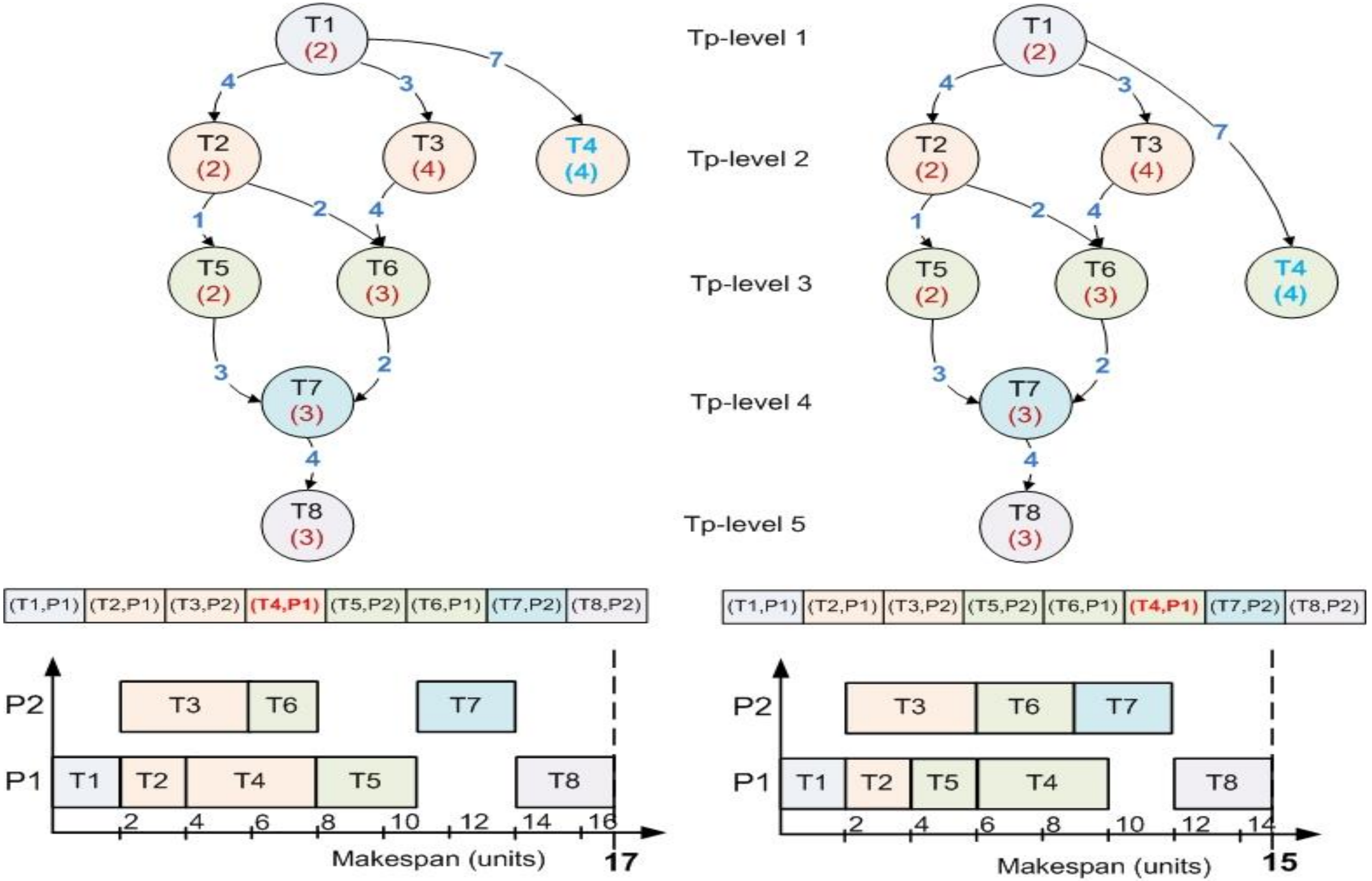
- **Population representation**
  - vector of fixed length representing the totality of chromosomes in a generation

Chromosome Encoding for DAG



(T1,P1)	(T2,P1)	(T3,P2)	(T4,P3)	(T5,P1)	(T6,P3)	(T7,P2)

Floating Nodes (DAG)



Genetic Operators (1)

- **Crossover**
  - type: single-point crossover
  - the processors are interchanged between genes in the same position of two different chromosomes

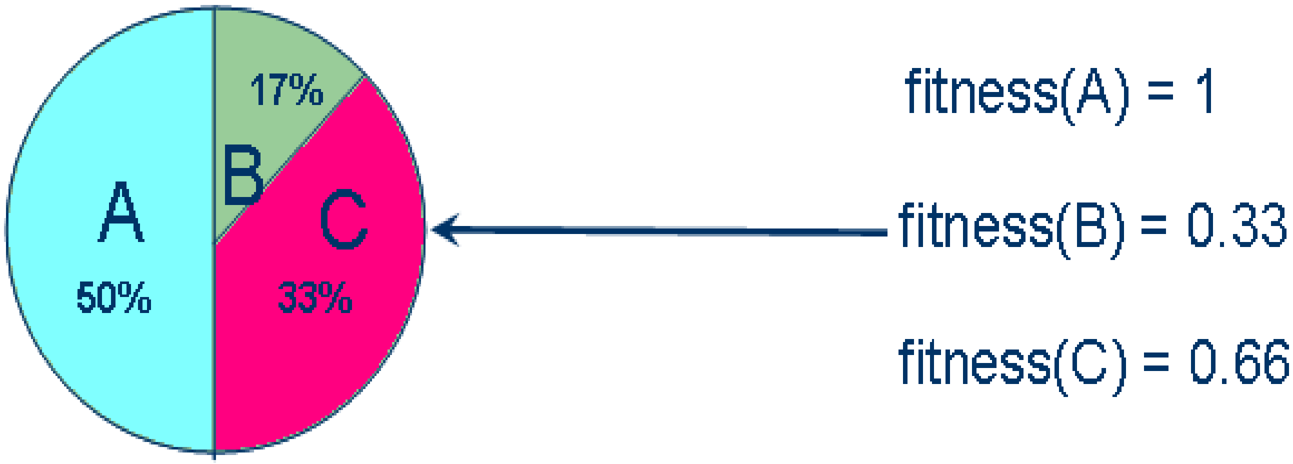
Parent 1	**(1,2)**	**(5,4)**	**(4,2)**	**(3,3)**	**(6,2)**	**(2,3)**
Parent 2	**(4,1)**	**(6,3)**	**(3,2)**	**(1,1)**	**(2,4)**	**(5,2)**
Offspring 1	**(1,2)**	**(5,4)**	**(4,2)**	**(3,1)**	**(6,4)**	**(2,2)**
Offspring 2	**(4,1)**	**(6,3)**	**(3,2)**	**(1,3)**	**(2,2)**	**(5,3)**

- **Mutation**
  - Alter a gene = assign a task on a different processor, randomly chosen
  - The search space is enlarged to the vicinity of the current population => tendency to converge to a global rather than to a local optimum
  - Pursued with a certain probability (mutation rate) for each chromosome
  - Adaptive mutation:
    - the mutation rate increases when the best solution for the population stagnates over a number of generations

Genetic Operators (2)

- **Selection**
  - roulette-wheel selection: each chromosome is assigned a survival probability according to its fitness contribution in the total population fitness:

$$prob(i) = \frac{F(i)}{F_t}$$



$$F_t = 1.00 + 0.33 + 0.66 \approx 2$$

Fitness Function (1)

- **The fitness function (for tasks without dependencies) applied in DIOGENES is:**

$$F = \left(\frac{t_m}{t_M}\right) \times \left(\frac{1}{n}\sum_{i=1}^{n}\frac{t_i}{t_M}\right) \times \left(\frac{T_s}{T}\right) \qquad 0 \le F \le 1$$

where:

n - the number of processors

t_i - the total execution time for processor *i.*

$t_m - min_{1<=i<=n}\{t_i\}$

$t_M - max_{1<=i<=n}\{t_i\}$

T_s - denotes the number of tasks which satisfy deadline and computation resource requirements.

T - represents the total number of tasks in the current schedule

Fitness Function (2)

$$F = \left(\frac{t_m}{t_M}\right) \times \left(\frac{1}{n}\sum_{i=1}^{n}\frac{t_i}{t_M}\right) \times \left(\frac{T_s}{T}\right)$$

The factor converges to 1 when t_m approaches t_M, and the schedule is perfectly balanced.

Average utilization of processors. In the ideal case, the total execution times on the processors are equal and equal to *maxspan*, which leads to a value of 1 for average processor utilization.

This factor acts like a contract penalty on the fitness. Its value varies reaching 1 when all the requirements are satisfied and decreases proportionally with each requirement that is not met.

Fitness Function for DAG

$$F(ch) = \left(\frac{t_m}{t_M}\right) \times \left(\frac{1}{n}\sum_{i=0}^{n}\frac{t_i}{t_M}\right) \times \left(\frac{T_S}{T}\right) \times \left(\frac{t_M}{SL(ch)}\right)$$

Load balancing

Average idle time

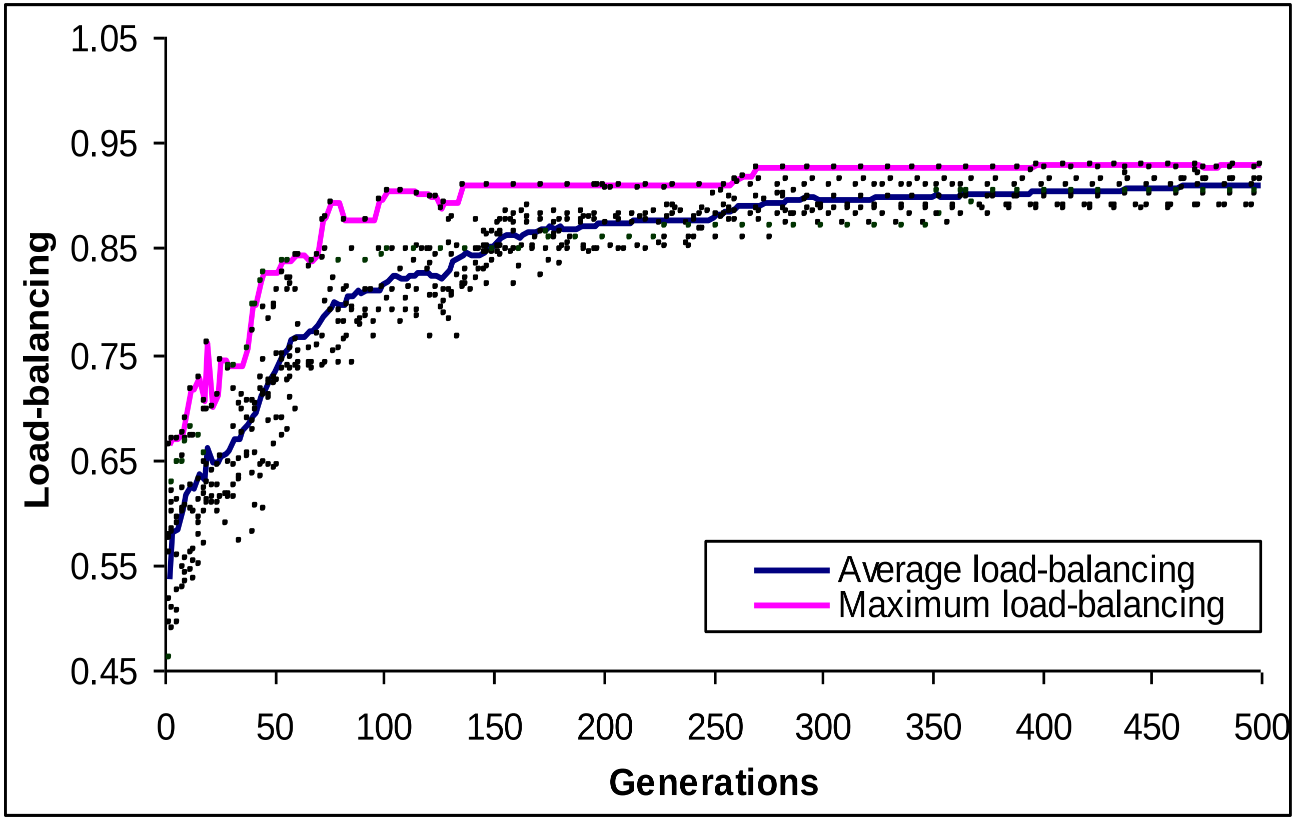
Deadline violation

Schedule length

Experimental Results

- **Experimental cluster configured with**
  - 11 (first phase) nodes and
  - 96 (second phase) nodes
- **The size of the task sets used for testing ranged**
  - between 50 and 100 tasks (first phase)
  - between 50 and 10000 tasks (second phase)
- **Parameters for the genetic algorithm**
  - crossover rate: 0.9
  - mutation rate: 0.005
  - chromosome length: 50

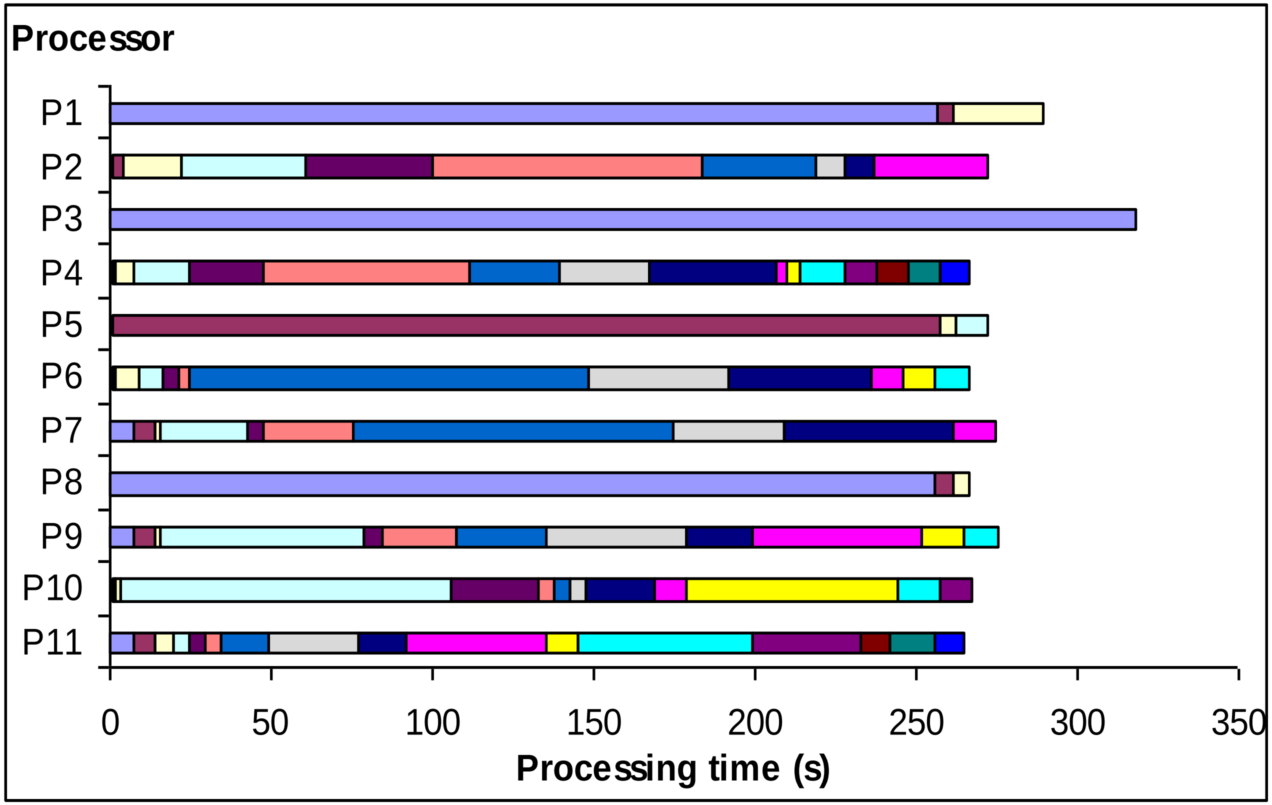
Algorithm Convergence



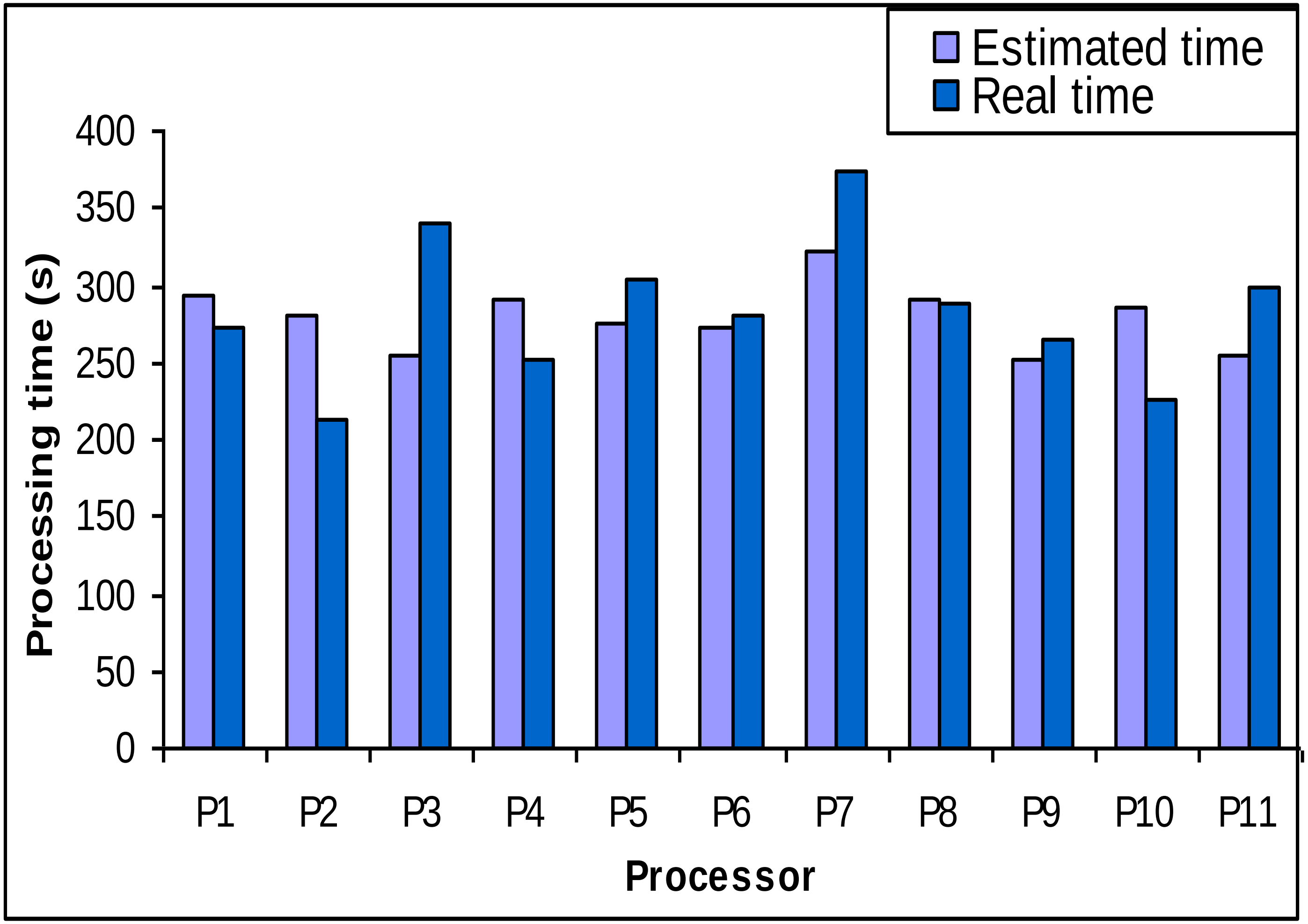
- The criterion used in convergence and algorithm stop condition is load-balancing between working nodes.
- The genetic algorithm converges after a small number of generation for various tests scenario.

Decentralization

- **Decentralized Cooperative Genetic Algorithm**
  - Parallel GA: multiple starting points, current best solutions exchange
  - Significant improvement of the solution quality
  - Total processing time, significantly reduced, Load-balancing



Estimate Times Versus Real Execution Times



$$\delta_i = \frac{t_i^r - t_i^e}{t_i^r}$$

$$\varepsilon = \sqrt{\frac{\sum_{i=1}^{n} \delta_i^2}{n}}$$

$$\varepsilon = 0.16$$

Comparison of Various Scheduling Methods

Algorithm	Load Balancing	Average Processor Utilization	Maxspan (s)
First Come First Serve	0.754	0.64	421.0
Centralized GA	0.752	0.70	388.0
Decentralized Non-Cooperative GA	0.780	0.74	369.5
Decentralized Cooperative GA	0.940	0.86	317.0

Conclusions

- Genetic scheduling algorithm for the problem of task allocation
- Improvement upon centralized genetic approaches with respect to scalability and robustness.
- Performances comparison for different scheduling strategies.
  - Decentralization and cooperation provide significantly better results of load-balancing and average processor utilization increase, as well as of total execution time minimization.
- Validation in real-time environments by utilizing existing monitoring and job execution systems. The experiments show a high level of accuracy in the results obtain
- ***…and*, “*Always searching to a better solution*”.**

Thank you!

GridMOSI (Virtual organization based on Grid Technology for High Performance Modelling, Simualtion and Optimization) – National Romania Project
http://gridmosi.ro

DIOGENES – Distributed near-Optimal Genetic Algorithm for Grid Applications Scheduling
http://diogenes.grid.pub.ro

Feature - Evolving towards the future of science: genetic algorithms and grid computing
http://www.isgtw.org/?pid=1000845