A Grid-oriented Genetic Algorithm

José Herrera Sanz
Eduardo Huedo Cuesta
Rubén Santiago Montero
Ignacio Martín Llorente

www.gridway.org

GW

Advanced Computing Laboratory
Associated to *NASA Astrobiology Institute*
CSIC-INTA

CSIC

INTA

Distributed Systems Architecture and Security Group
Universidad Complutense de Madrid

Outline

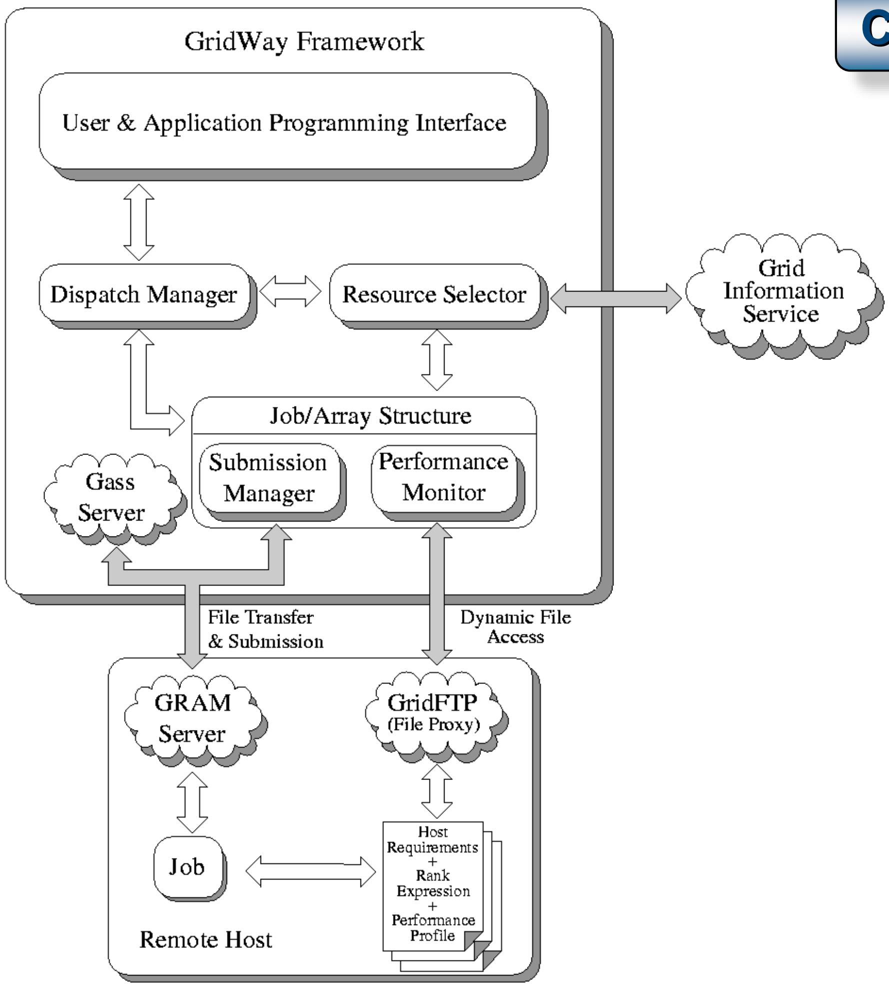
- Motivation
- The Grid*W*ay Framework
- DRMAA: Distributed Resource Management Application API
- Development Model
- Parallel Genetic Algorithms
- Algorithm Description
- Experimental Results
- Conclusions

Motivation

- **GAs** are stochastic search methods that have been successfully applied in many search, optimization, and machine learning problems.
- **PGAs** offers many advantages over the traditional GAs (speed, wok in a larger search space, and less likely to run into a local optimum).
- With the advent of Grid computing, the computational power that can be deliver to the applications have substantially increased.
- **PGAs** can potentially benefit from this new **Grid technologies**.
- Implementation and execution of **PGAs** in a Grid involve challenging issues.
- **Our research**: **PGA** across the **Grid** using the **DRMAA** standard **API** and the **Grid*Way*** framework.
- The **efficiency** and **reliability** of before schema to solve the **One Max** problem is analyzed in a **globus-based** research **testbed**.

The GridWay Framework

GridWay provides an easier and more efficient execution (***submit & forget***) on **heterogeneous** and **dynamic** Grid.



Characteristics

- **Dynamic Scheduler**: GridWay periodically adapts the scheduler to the available resources
- **Resource Selector**: Reflects the applications demands, in terms of requirements and preferences.
- **Adaptive Job Execution**: To migrate running applications to more suitable resources.
- **Fault tolerance** (callbacks) and **Job exit codes** (Job-manager).

DRMAA

Distributed Resource Management Application API

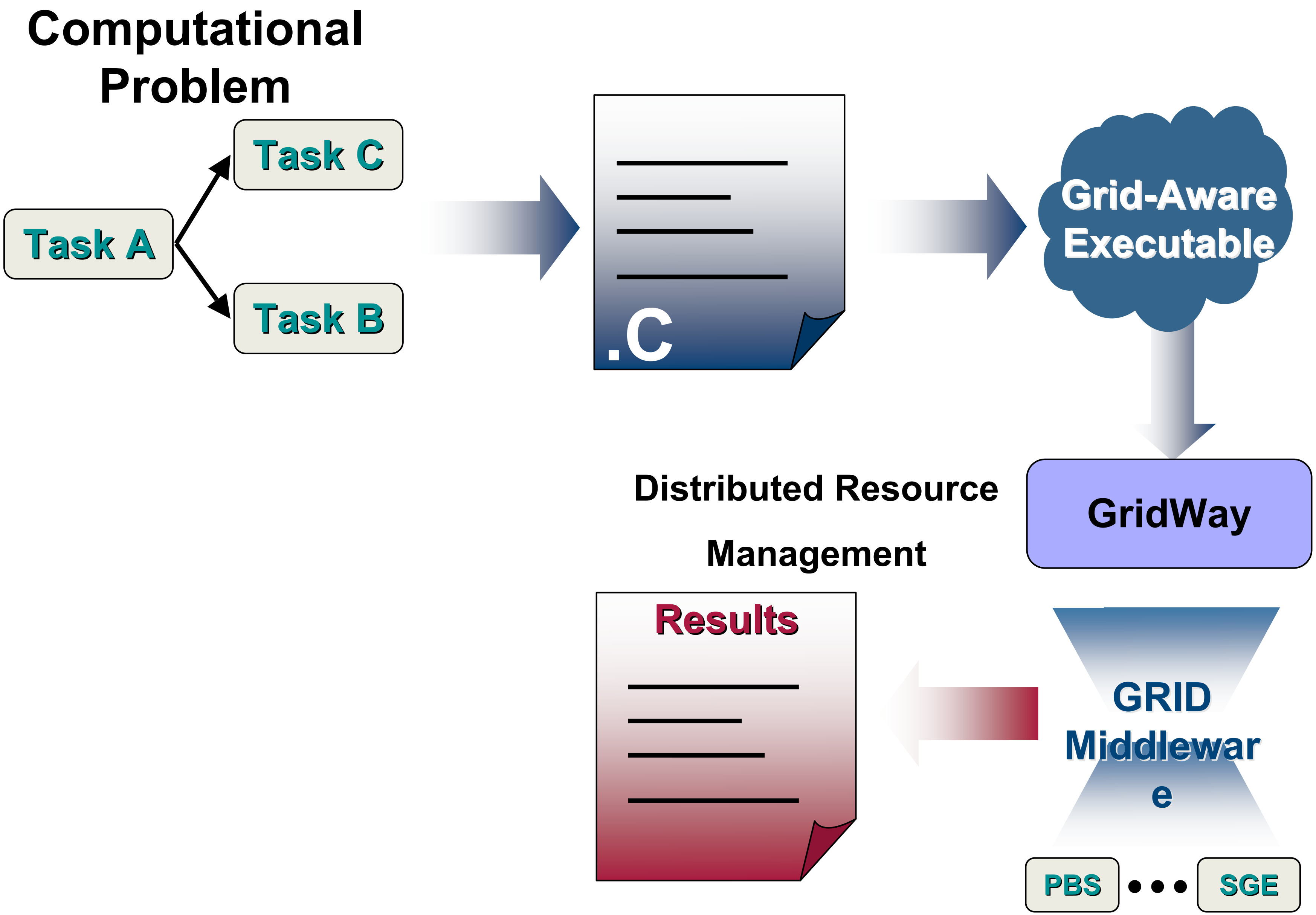
- The DRMAA specification constitutes a **homogenous interface** to different **DRMS** to handle job submission, monitoring and control, and retrieval of finished job status. Moreover, DRMAA has been developed by DRMAA-WG within the Global Grid Forum (GGF).

- The DRMAA standard represents a suitable and portable framework to express this kind of distributed computations.

- Some DRMAA interface routines:
  - Initialization and finalization routines: **`drmaa_init`** and **`drmaa_exit`**.
  - Job submission routines: **`drmaa_run_job`** and **`drmaa_run_bulk_jobs`**.
  - Job control and monitoring routines: **`drmaa_control`**, **`drmaa_synchronize`**, **`drmaa_wait`** and **`drmaa_job_ps`**.

- DRMAA interface routines has been implemented within the **GridWay** framework.

Development Model



Parallel Genetic Algorithms (1/2)

Single Population (Panmitic GA)

- Usually implemented using a **Master/Worke**r paradigm.
- Can be efficiently used when evaluation function requires a considerable amount of computational work.
- **Main advantage**: the search behavior of the **sequential GA** is not altered.
- **Disadvantage**: This approach **is not well** suited for a **Grid** because of the high network requirements of its communication patters.

Single Population (Fine Grain GA)

- Only **one population** and its spatial structure limits the interactions between individuals.
- This limit can be imposed:
  - **Chromosome** level: **each member** can only interact with their **neighbors**
  - **Population** level: **only member** of the **same subpopulation** may mate during crossover.

Parallel Genetic Algorithms (2/2)

Coarse Grain GA

- Main population is divided into **subpopulations** (demes) each one independently **evaluated** in a **different** node.

- **Tree** possible communication patters:
  - **Ring model**: processes can only interact with their neighbors in a ring topology
  - **Master-slave model**: slave processes swap best individuals with the master.
  - **All-to-all model**: All processes swap best individuals with the others.

- **Disadvantage**: Introduce fundamental **changes** in the implementation of a **simple GA**.
- **Advantage**: It is **more tolerant** to the **high latencies** and **dynamic bandwidths** that can be expected in the **Internet**, unlike the **single population** alternatives.

- In this **research** we use a modified version of the **coarse grain** approach, since this algorithm does **not imply a tightly coupled** deme topology

Algorithm Description

Main Characteristics

- We use a **fully connected multi-deme** genetic algorithm, **all demes exchange** individuals every generation.
- **Not** imply any **overhead** since the population of **each deme** is used as **checkpoint** files.

Algorithm Execution

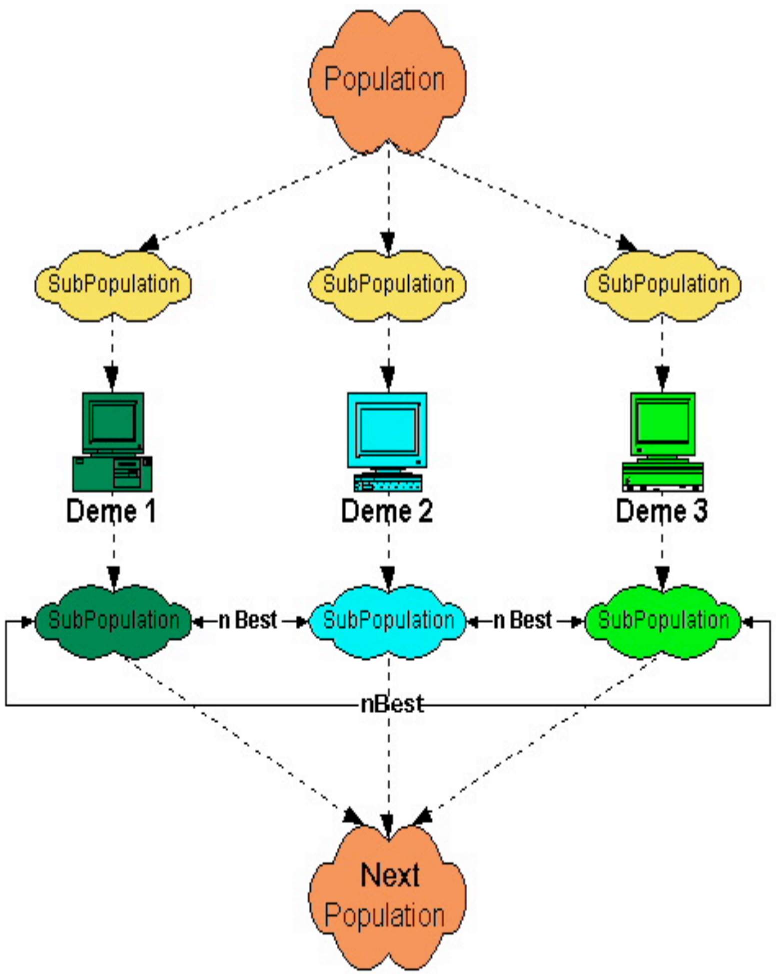
- Initial **population** is uniformity **distributed** among available number of **nodes**.
- Sequential **GA** is locally **executed** over each **subpopulations**.
- **Worst** individuals of each **subpopulation** are exchanged with the **best** ones of the rest.
- **New** population is **generated** to perform the **next** iteration.

Algorithm Optimization

- Previous algorithm may incur in **performance losses**, since the **iteration time** is determined by the **slowest machine**.
- **Solution** ⇨ *Dynamic Connectivity:*
  - We allow an asynchronous communication pattern between a fixed number of demes.
  - Minimum number of demes in each iteration depends on the numerical characteristics of the problem.

Algorithm Schema

Algorithm Schema



DRMAA Schema

```
// Initialize a new DRMAA session.
rc = drmaa_init(contact, error);
// Execute all jobs consecutively
for (i=0; i < ALL_JOBS; i++)
  rc = drmaa_run_job(job_id, jt,
       err_diag);
// Execute GOGA if it doesn't rise
// objetive_function
while (!this->objetive_function()){
  // Wait for (dynamic conectivity
  // degree) jobs and store results
  for (i=0; i < NUM_JOBS; i++)
    rc = drmaa_wait(job_id, &stat,
         timeout, rusage, err);
  this->store_results();
  // Execute (dynamic connectivity
  // degree) jobs consecutively
  for (i=0; i < NUM_JOBS; i++)
    rc = drmaa_run_job(job_id, jt,
                       err);}
// Finalize DRMAA session.
rc = drmaa_exit(err_diag);
```

Experiences (1/5)

Testbed description

Host	Model	Hz	OS	Memory	Nodes	GRAM
babieca	Alpha DS10	466Mhz	Linux 2.2	256MB	5	PBS
hydrus	Intel Pentium 4	2.5 Ghz	Linux 2.4	512MB	1	fork
cygnus	Intel Pentium 4	2.5 Ghz	Linux 2.4	512MB	1	fork
aquila	Intel Pentium III	666 Mhz	Linux 2.4	128 MB	1	fork

Objectives

- We evaluate the functionality and efficiency of the Grid-oriented Genetic Algorithm, in the solution of the **One-Max problem**.
- One-Max problem is a **classical benchmark** problem for genetic algorithm computations, and it tries to evolve an initial matrix of **zeros** in a matrix of **ones**.

Experiences (2/5)

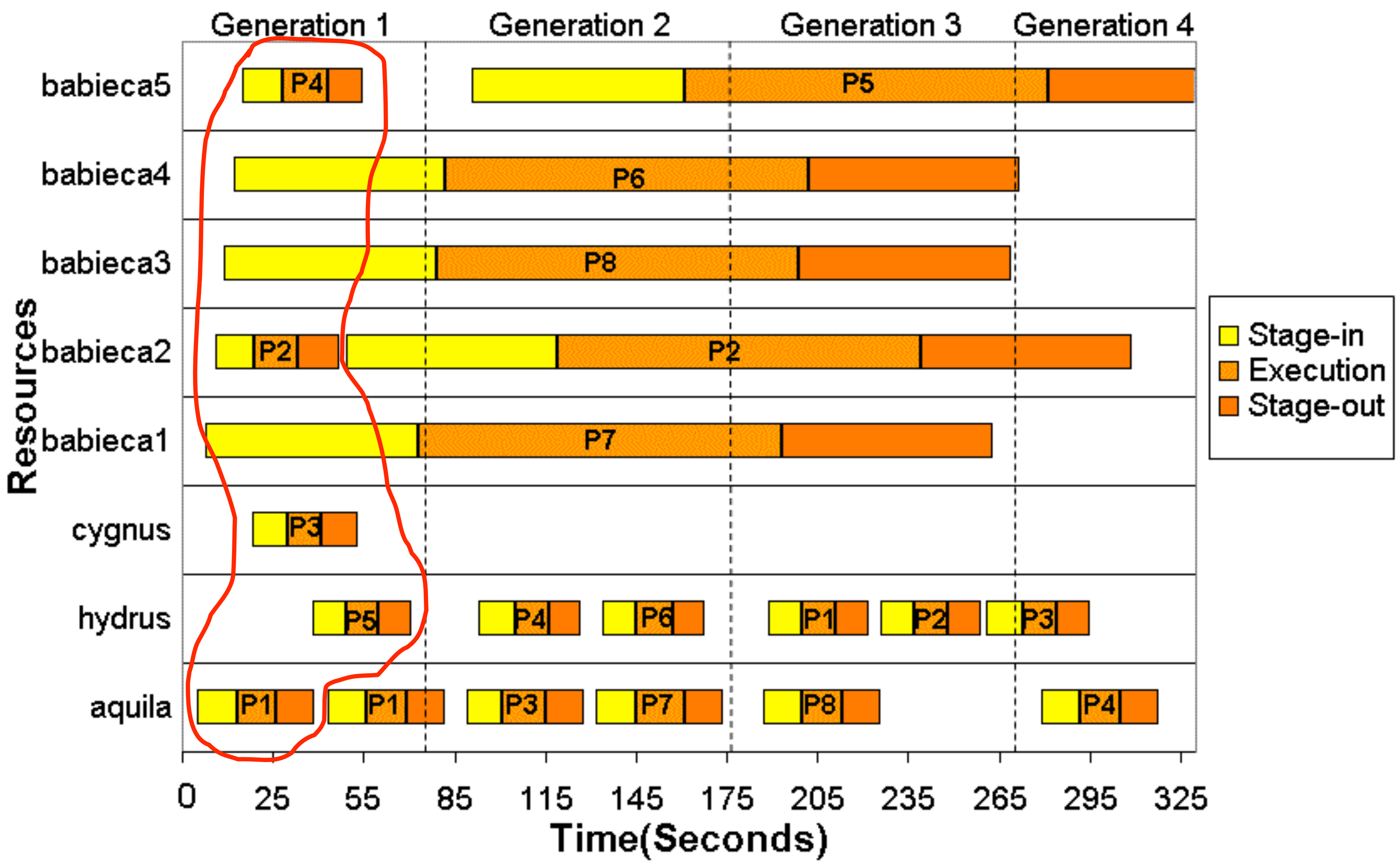
Experience description

- We will consider an initial population of **1000** individuals each one a **20x100** zero matrix.
- Sequential GA:
  - Iterations: 50
  - Mutation Probability: 0.1%
  - Crossover Probability: 60%
- The exchange probability of best individuals between demes is **10%**
- Final score must be a **1800** ones **matrix**.

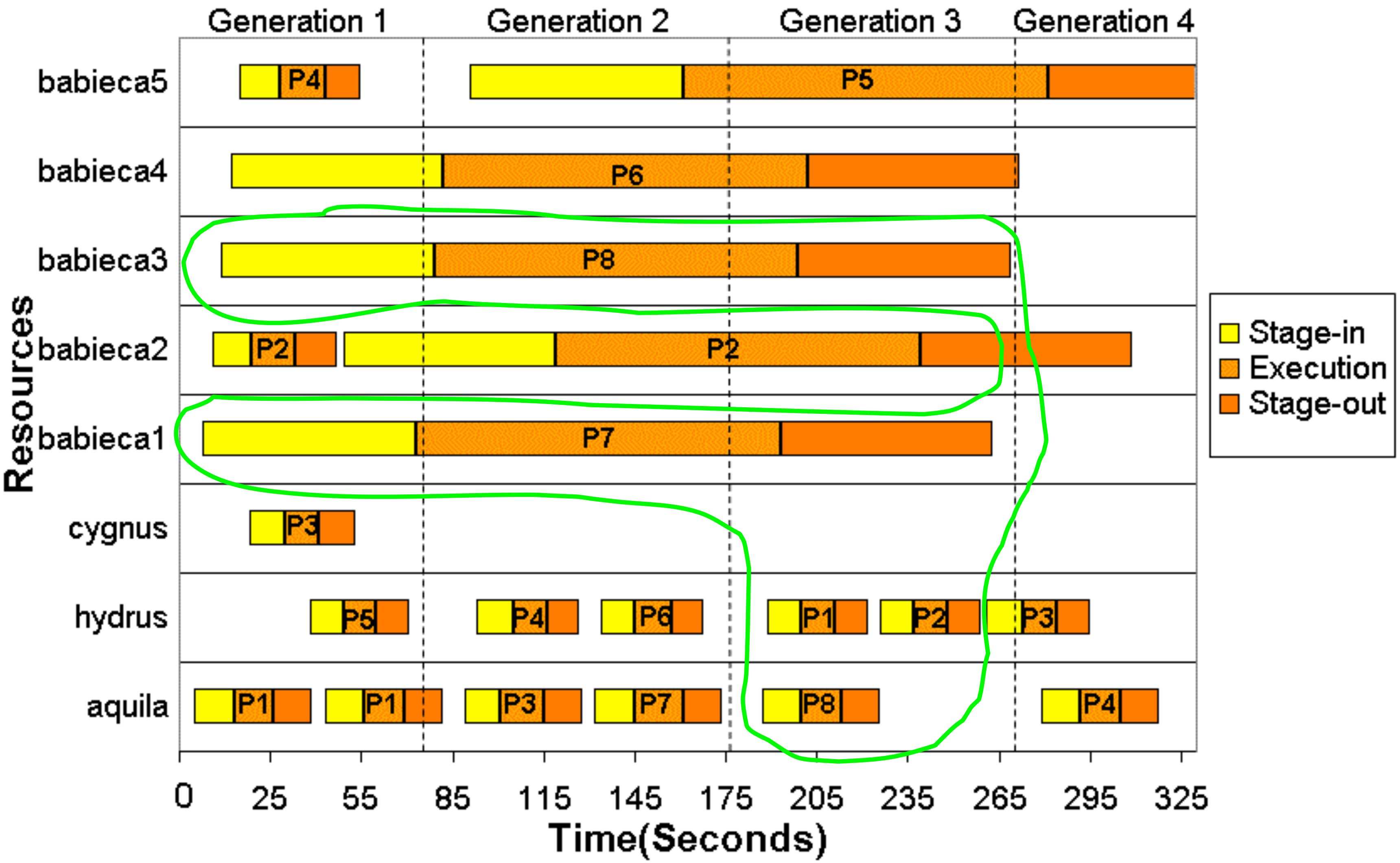
Two experiments

- Execution profile of **4 generations** of the **GOGA**, with a **5-way** *dynamic connectivity*.
- **5 different** executions of **GOGA**, with **different** degrees of *dynamic connectivity*: **2-way**, **3-way**, **5-way**, **6-way** and **8-way**.

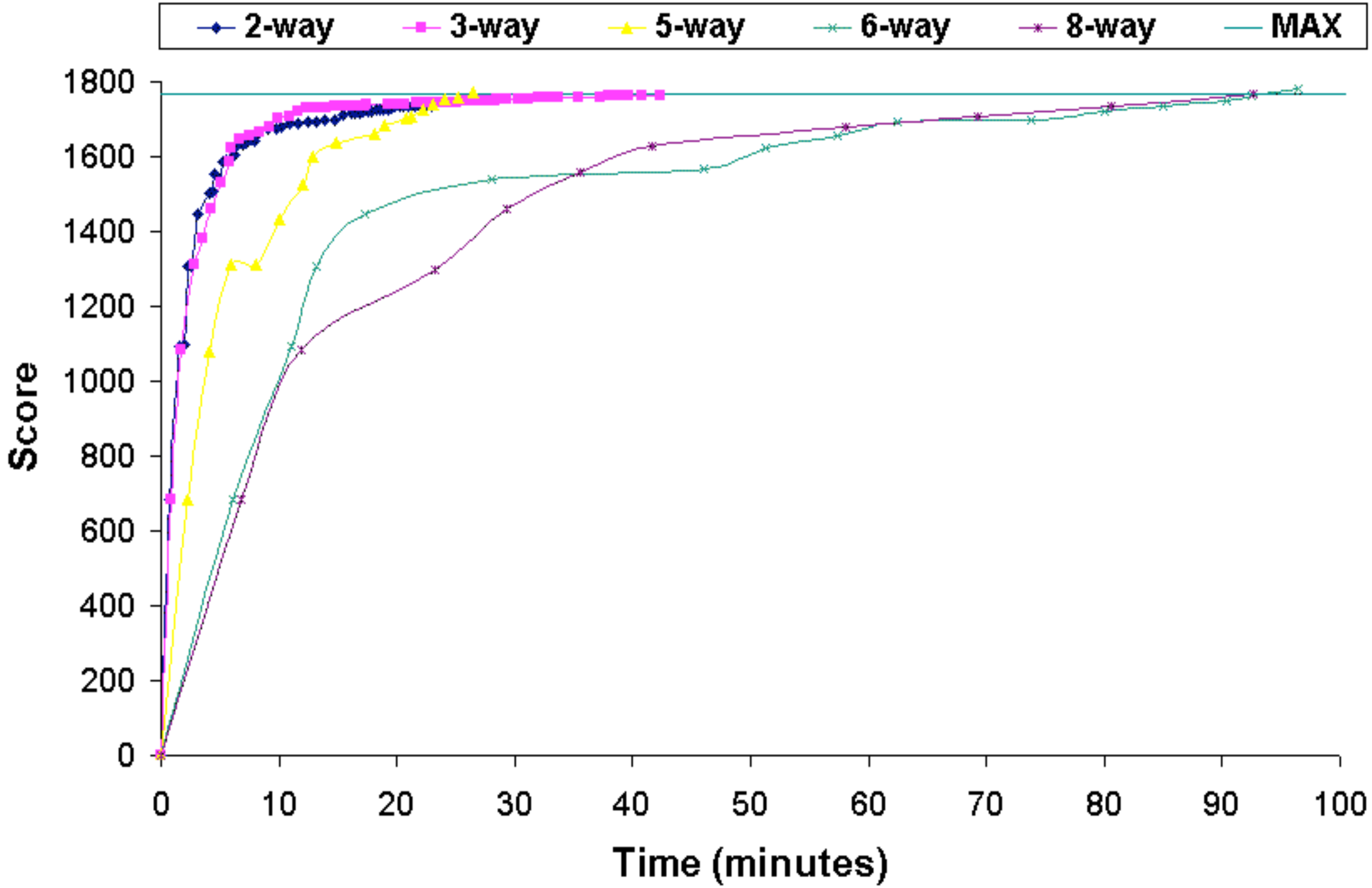
Experiences (3/5)



Experiences (4/5)



Experiences (5/5)



Conclusions

- We have presented an efficient **Grid-oriented genetic algorithm**.
- Our approach uses a **fully connected multi-deme** GA, with a ***dynamic connectivity*** between subpopulations to deal with the **heterogeneity of the Grid**.
- The **optimum degree** connectivity depends on:
  - The computational **characteristics** of the **Grid nodes**.
  - The computational **problem**.
- The **GOGA** has been developed taking advantage of the **GridWay** framework features and the **DRMAA API**.
- It have been shown that **DRMAA** can aid the rapid development and **distribution** across the **Grid** of typical genetic algorithm strategies.

A Grid-oriented Genetic Algorithm

José Herrera Sanz
Eduardo Huedo Cuesta
Rubén Santiago Montero
Ignacio Martín Llorente

www.gridway.org

Advanced Computing Laboratory
Associated to *NASA Astrobiology Institute*
CSIC-INTA

Distributed Systems Architecture and Security Group
Universidad Complutense de Madrid