

New Self-Scheduling Schemes for Internet-Based Grids of Computers

Javier Díaz

javier.diaz@uclm.es

Sebastián Reyes, Alfonso Niño, Camelia
Muñoz-Caro



Grupo de Química Computacional y Computación
de Alto Rendimiento (QCyCAR)
Escuela Superior de Informática
Universidad de Castilla-La Mancha

Ciudad Real. España



Outline

- **Motivation**
- Background
- New Schemes
- Methodology
- Tests Results
- Conclusions

Motivation

- Scheduling of tasks at the application level in an Internet-based Grid of computers
- The Grid is an heterogeneous and geographically distributed environment
- Allocate efficiently workload
- Balance between computer loads and communication overheads
- Self-Scheduling algorithms are extensively used

Outline

- Motivation
- **Background**
- New Schemes
- Methodology
- Tests Results
- Conclusions

Self-Scheduling Algorithms

- A class of adaptative/dynamic centralized loop scheduling algorithms
- Allocate sets of tasks (chunks) among processors
- Assume that the duration of the tasks are different and not known
- Try to find an equilibrium between load balance and overhead
- A number of variants has been proposed using different formulates for the chunk sizes

Chunk Self-Scheduling (CSS)

- Chunk size determined by the programmer
- Difficult to fix an optimal chunk size
- A large chunk size will cause load imbalancing
- A small chunk size is likely to produce too much scheduling overhead

Guided Self-Scheduling (GSS)

- Chunk sizes are dynamically calculated

$$C_i = \frac{R_{i-1}}{P}$$

- Sometimes good load balance
- Small scheduling overhead

Trapezoid Self-Scheduling (TSS)

- Linearly decreasing chunk function

$$C_i = C_{i-1} - D$$

- Decreasing ratio (D) and Number of chunks (N):

$$D = \frac{F - L}{N - 1} \quad N = \left\lceil \frac{2 * I}{F + L} \right\rceil$$

- The developers of this algorithm proposed assign $F = I/2P$ and $L=1$ tasks to the first and last chunk, respectively
- Tries to reduce the need for synchronization while maintaining a reasonable load balance

Factoring Self-Scheduling (FSS)

- Results from a statistical analysis
- Tasks are scheduled in batches (or stages) of P chunks of equal size

$$C_i = \frac{R_{i-1}}{\alpha P}$$

- Only a subset of the remaining tasks is scheduled in each batch
- Better load balance than GSS when execution time changes widely and randomly

Self-Scheduling Algorithms Problems

- Derived for homogeneous system
- Not enough flexible to adapt efficiently to a heterogeneous environment
- Do not developed for a geographically distributed environment

Self-Scheduling Algorithms for Heterogeneous Systems

- Algorithms based on weight factors
 - **Weighted Factoring (WF)**. A weight factor for each processor
 - **Adaptive Factoring (AF)**. Variable weight factor depending on variations in the computation power
 - **Distributed TSS (DTSS)**. Weights calculated using the relative power and the number of processes in their run-queue
- Algorithms based on historic records
 - **Self-Adapting Scheduling (SAS)**. History of task timing results.
- Algorithm for dependence loops
 - **DTSS+SP**. Handle loops with dependencies on heterogeneous clusters via synchronization points
- Other algorithms are **Gap-Aware Self-Scheduling (GAP)** and **RAS**

Outline

- Motivation
- Background
- **New Schemes**
- Methodology
- Tests Results
- Conclusions

Introduction

- The chunk function $C(t)$ can be expressed by a Taylor expansion

$$C(t) = f(t_0) + \left. \frac{df(t)}{dt} \right|_0 (t - t_0) + \frac{1}{2} \left. \frac{d^2 f(t)}{dt^2} \right|_0 (t - t_0)^2 + \dots$$

$$C(t) = a + bt + ct^2 + \dots$$

- Limiting the expansion to the constant term
 - Pure self-scheduling (SS) or Chunk Self-Scheduling (CSS)
- Limiting the expansion to the linear term
 - Trapezoid Self-Scheduling (TSS)

Quadratic Self-Scheduling

- Based in the chunk distribution function
- A more flexible model is obtained if we retain up to the quadratic term
 - Quadratic Self-Scheduling (QSS)

$$C(t) = a + bt + ct^2$$

- Three reference points $(C(t), t)$
 - The first point as the linear case $(C_0, 0)$. $C_0 = I/2P$
 - (C_N, N) and $(C_{N/2}, N/2)$ are experimentally determined. $C_{N/2} \in (C_0, C_N)$

$$a = C_0$$

$$b = (4C_{N/2} - C_N - 3C_0)/N$$

$$c = (2C_0 + 2C_N - 4C_{N/2})/N^2$$

$$N = 6I/(4C_{N/2} + C_N + C_0)$$

$$C_{N/2} = \frac{C_0 + C_N}{\delta}$$

Exponential Self-Scheduling

- Based in the slope of the chunk distribution function

$$\frac{dC(t)}{dt} = f(t)$$

- A first approach is to consider that the slope (negative) is proportional to the chunk size

$$\frac{dC(t)}{dt} = -kC(t)$$

- The $C(t)$ function is obtained integrating

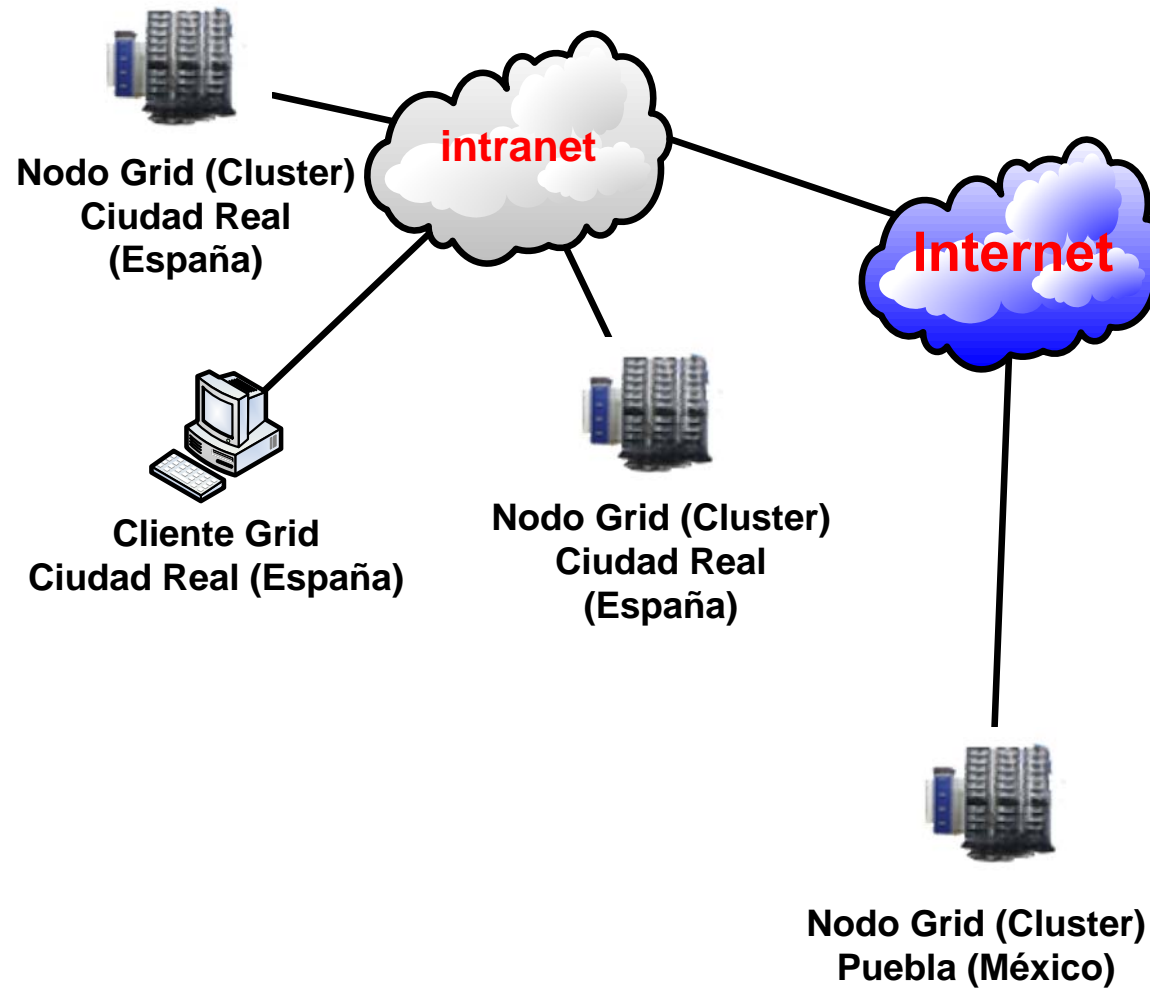
$$C(t) = \left(\frac{I}{2P} \right) e^{-kt}$$

- $C(0) = I/2P$
- k is experimentally determined

Outline

- Motivation
- Background
- New Schemes
- **Methodology**
- Tests Results
- Conclusions

Topology



Tests Performed

- Comparison of QSS and ESS against CSS, GSS, TSS and FSS
- Case performed (Tasks/Processors):
 - 2804/20
- Sets of tasks without any predefined relationship among their durations

Algorithms Configuration

- CSS. Each chunk is I/P tasks
- GSS. Chunk is R_{i-1}/P tasks
- TSS. $F = I/2P$ and $L=1$
- FSS. $\alpha = 2$
- QSS. $C_0 = I/2P$, C_N and $C_{N/2}$ are experimentally determined
- ESS. $C_0 = I/2P$, k is experimentally determined

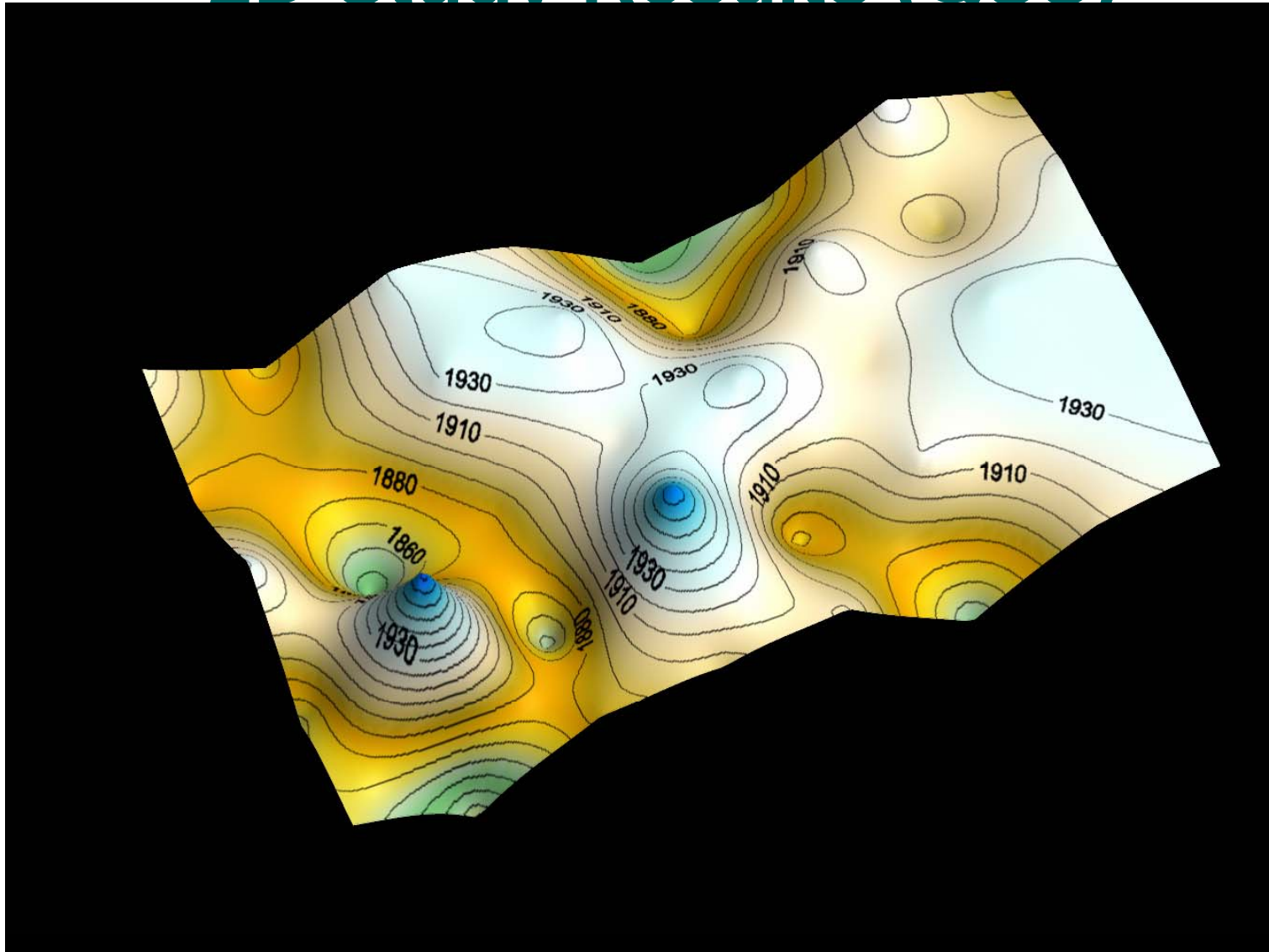
Outline

- Motivation
- Background
- New Schemes
- Methodology
- **Tests Results**
- Conclusions

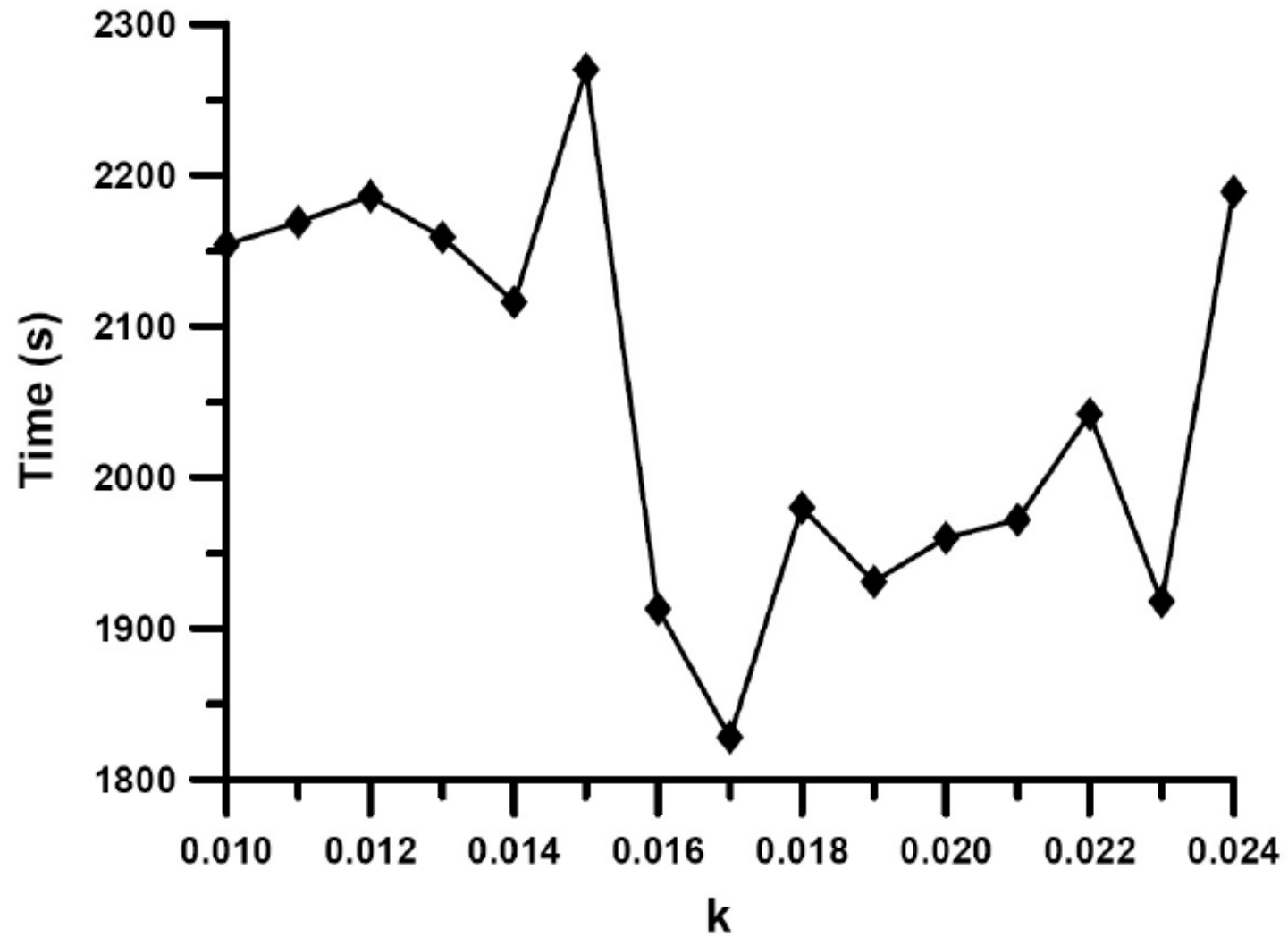
Test Results

- 2D study to obtain the optimal values of C_N and $C_{N/2}$ parameters for the QSS algorithm
- Calculate optimal value of k parameter for the ESS algorithm
- Speedup of different algorithms

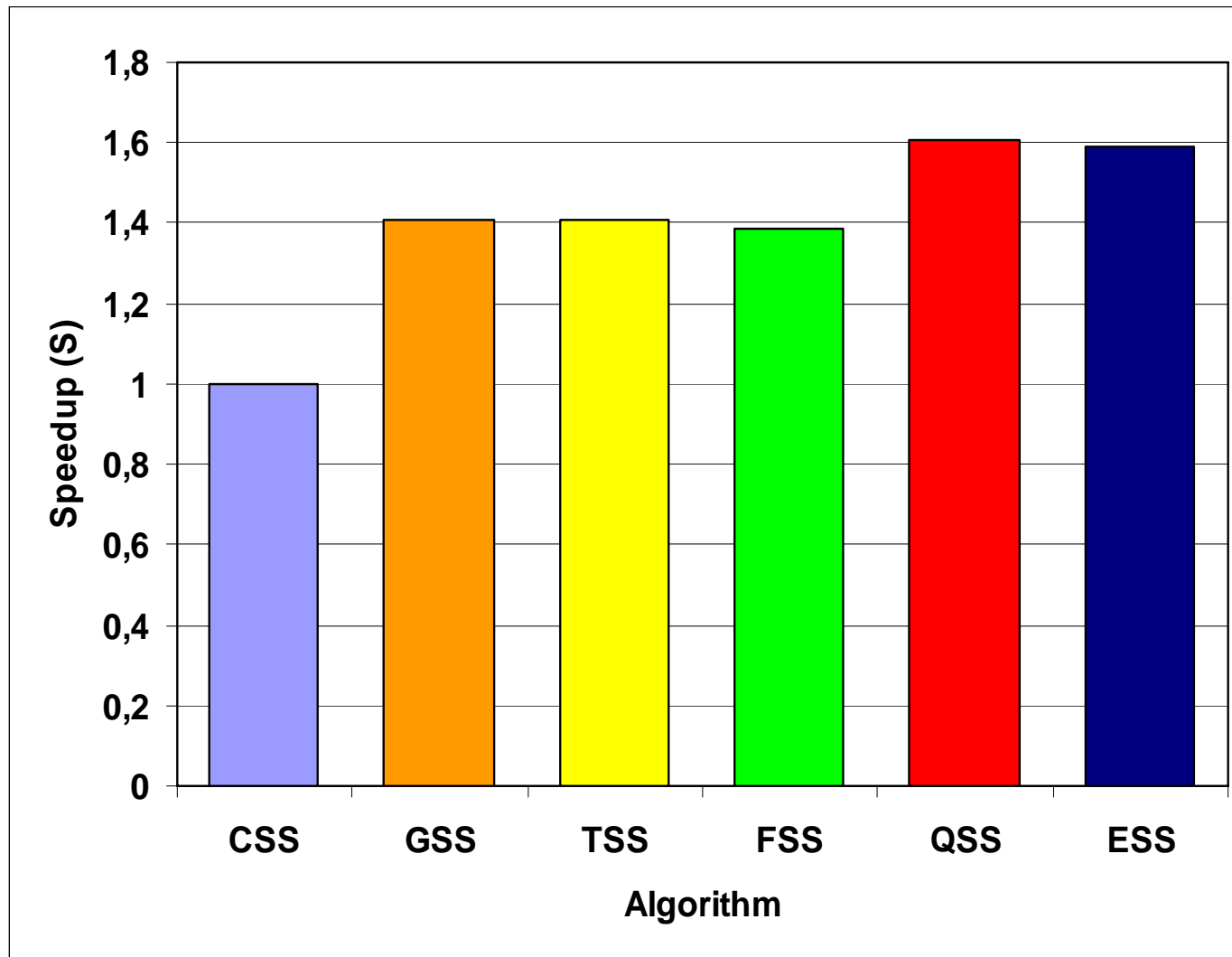
2D Study Results (QSS)



k Results (ESS)



Speedup



Outline

- Motivation
- Background
- New Schemes
- Methodology
- Tests Results
- **Conclusions**

Conclusions

- Self-Scheduling algorithms can be used in heterogeneous and distributed systems, but they are not sufficiently adaptive
- Quadratic Self-Scheduling algorithm has more degrees of freedom than the previous ones
- Exponential Self-Scheduling algorithm has similar performance than QSS with only two parameters
- Both, QSS and ESS outperform all the other self-scheduling algorithms

Aknowledgements

- **Organisms**

- **Junta de Comunidades de Castilla-La Mancha** (*grant # PBI05-009*)
- **Ministerio de Educación y Ciencia** (*grant # PBI05-009*)
- **Universidad de Castilla-La Mancha**
- **Departamento de Tecnologías y Sistemas de Información de la Escuela Superior de Informática de la UCLM**

- **Institutions**



Thanks for your attention!

<http://qcycar.inf-cr.uclm.es>

