Universidad de La Laguna

ESCUELA TÉCNICA SUPERIOR
INGENIERÍA INFORMÁTICA

Solving simulation optimization problems on grid computing systems

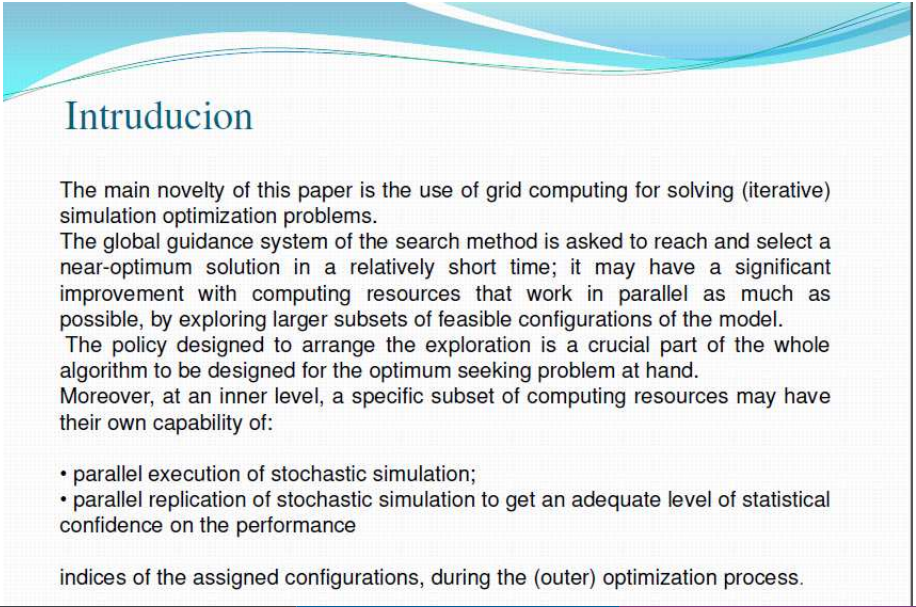
YOUSEF RAJAEI TABRIZI

Universidad de La Laguna

11 de enero de 2016

Escuela Superior de Ingeniería y Tecnología
Universidad de La Laguna

Generate model



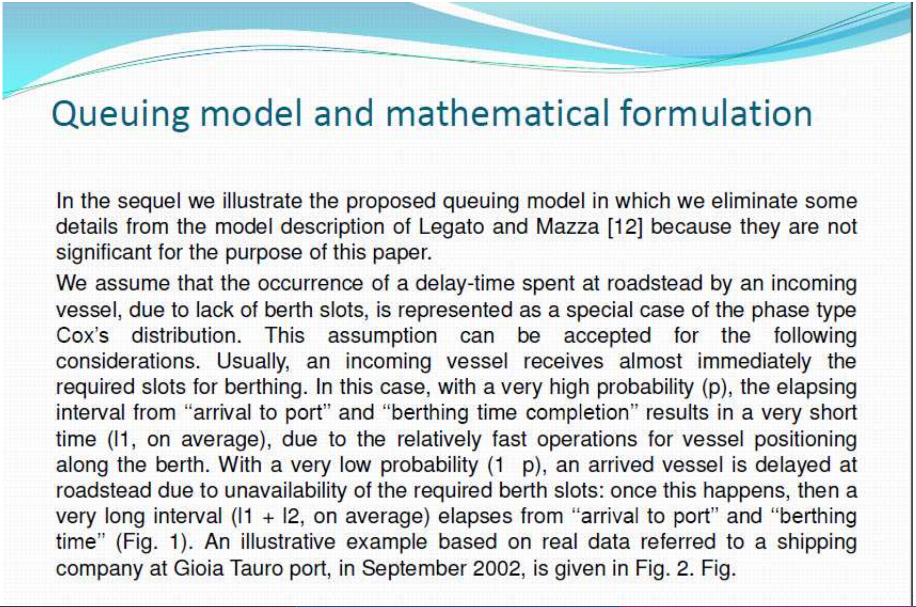
Intruducion

The main novelty of this paper is the use of grid computing for solving (iterative) simulation optimization problems.
The global guidance system of the search method is asked to reach and select a near-optimum solution in a relatively short time; it may have a significant improvement with computing resources that work in parallel as much as possible, by exploring larger subsets of feasible configurations of the model.
The policy designed to arrange the exploration is a crucial part of the whole algorithm to be designed for the optimum seeking problem at hand.
Moreover, at an inner level, a specific subset of computing resources may have their own capability of:

• parallel execution of stochastic simulation;
• parallel replication of stochastic simulation to get an adequate level of statistical confidence on the performance

indices of the assigned configurations, during the (outer) optimization process.

The frequence of one exprimente



Queuing model and mathematical formulation

In the sequel we illustrate the proposed queuing model in which we eliminate some details from the model description of Legato and Mazza [12] because they are not significant for the purpose of this paper.

We assume that the occurrence of a delay-time spent at roadstead by an incoming vessel, due to lack of berth slots, is represented as a special case of the phase type Cox's distribution. This assumption can be accepted for the following considerations. Usually, an incoming vessel receives almost immediately the required slots for berthing. In this case, with a very high probability (p), the elapsing interval from "arrival to port" and "berthing time completion" results in a very short time (l1, on average), due to the relatively fast operations for vessel positioning along the berth. With a very low probability (1 p), an arrived vessel is delayed at roadstead due to unavailability of the required berth slots: once this happens, then a very long interval (l1 + l2, on average) elapses from "arrival to port" and "berthing time" (Fig. 1). An illustrative example based on real data referred to a shipping company at Gioia Tauro port, in September 2002, is given in Fig. 2. Fig.

Qeueing model

The queuing model to be optimized by repeated simulations

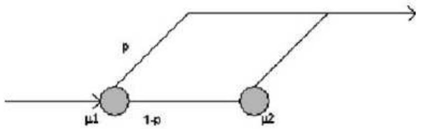
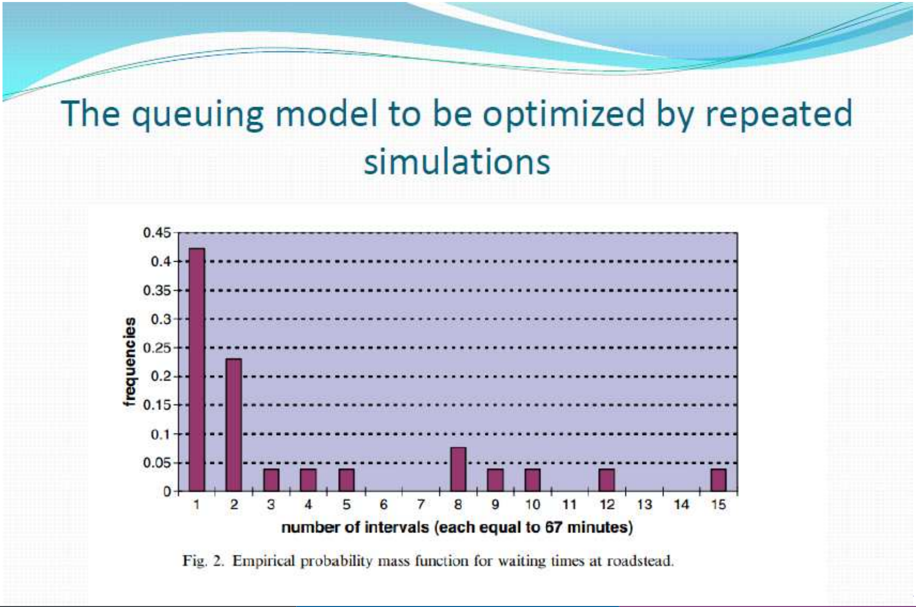


Fig. 1. Two phases, Coxian distribution.

Data table of qeueing model



The queuing model to be optimized by repeated simulations

Fig. 2. Empirical probability mass function for waiting times at roadstead.

The queuing model

The queuing model to be optimized by repeated simulations

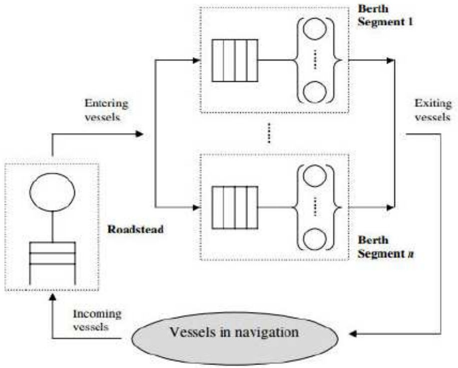
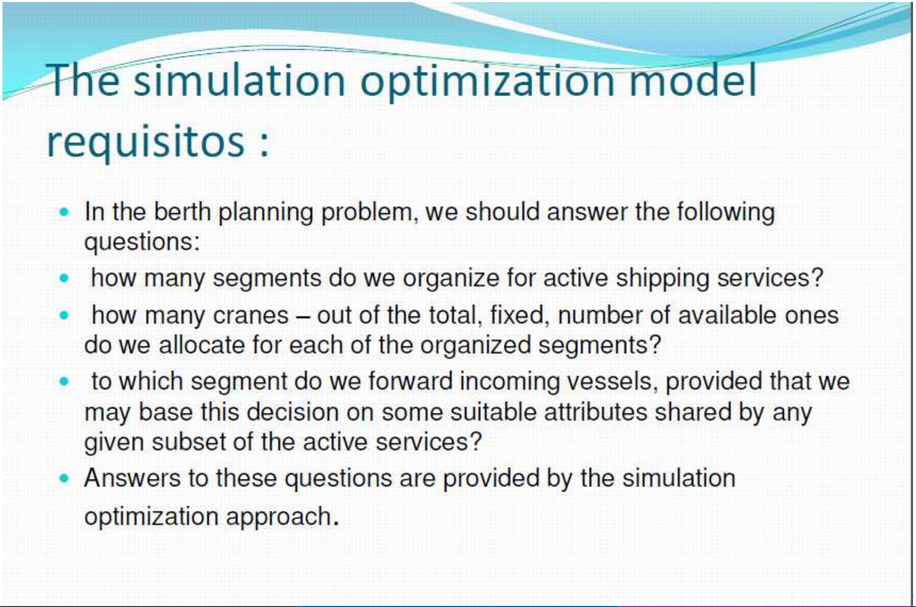


Fig. 3. The queuing network model for the logistic process under study

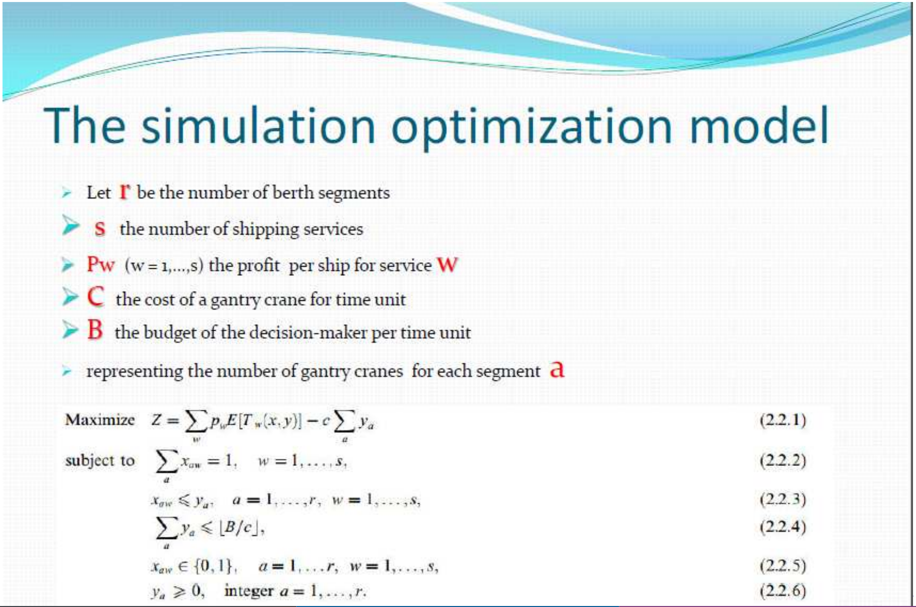
Requisitos



The simulation optimization model requisitos :

- In the berth planning problem, we should answer the following questions:
- how many segments do we organize for active shipping services?
- how many cranes – out of the total, fixed, number of available ones do we allocate for each of the organized segments?
- to which segment do we forward incoming vessels, provided that we may base this decision on some suitable attributes shared by any given subset of the active services?
- Answers to these questions are provided by the simulation optimization approach.

The formul generation



The simulation optimization model

- Let r be the number of berth segments
- s the number of shipping services
- Pw (w = 1,...,s) the profit per ship for service W
- C the cost of a gantry crane for time unit
- B the budget of the decision-maker per time unit
- representing the number of gantry cranes for each segment a

$$\text{Maximize} \quad Z = \sum_w p_w E[T_w(x, y)] - c \sum_a y_a \qquad (2.2.1)$$

$$\text{subject to} \quad \sum_a x_{aw} = 1, \quad w = 1, \ldots, s, \qquad (2.2.2)$$

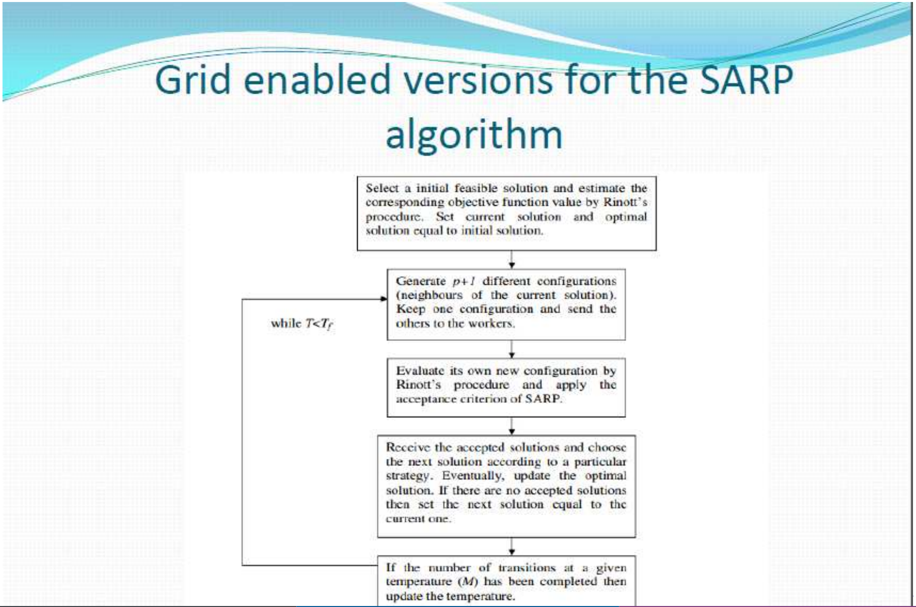
$$x_{aw} \leqslant y_a, \quad a = 1, \ldots, r, \quad w = 1, \ldots, s, \qquad (2.2.3)$$

$$\sum_a y_a \leqslant \lfloor B/c \rfloor, \qquad (2.2.4)$$

$$x_{aw} \in \{0, 1\}, \quad a = 1, \ldots, r, \quad w = 1, \ldots, s, \qquad (2.2.5)$$

$$y_a \geqslant 0, \quad \text{integer } a = 1, \ldots, r. \qquad (2.2.6)$$

Sharp algorithm



Grid enabled versions for the SARP algorithm

Select a initial feasible solution and estimate the corresponding objective function value by Rinott's procedure. Set current solution and optimal solution equal to initial solution.

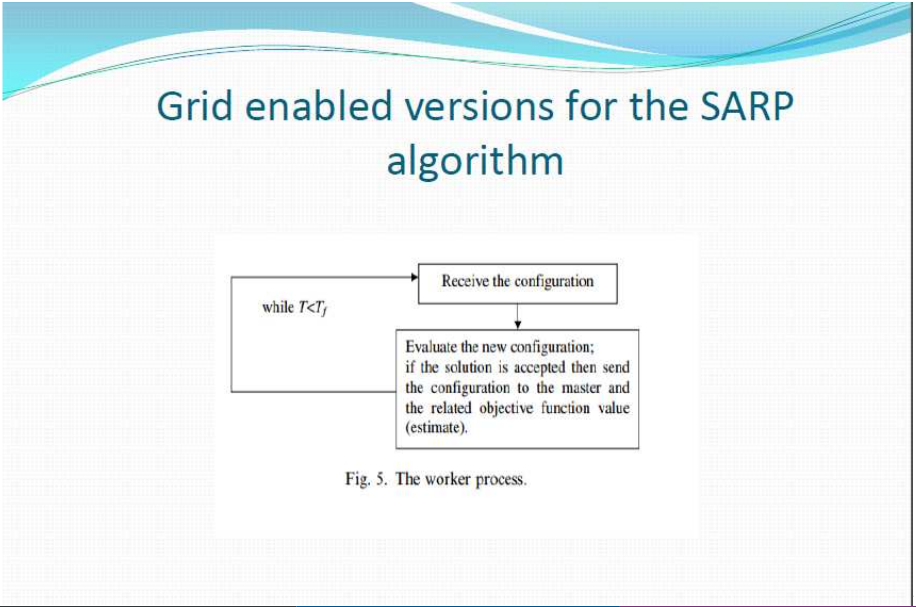
Generate $p+1$ different configurations (neighbours of the current solution). Keep one configuration and send the others to the workers.

Evaluate its own new configuration by Rinott's procedure and apply the acceptance criterion of SARP.

Receive the accepted solutions and choose the next solution according to a particular strategy. Eventually, update the optimal solution. If there are no accepted solutions then set the next solution equal to the current one.

If the number of transitions at a given temperature (M) has been completed then update the temperature.

Sharp algorithm



Grid enabled versions for the SARP algorithm

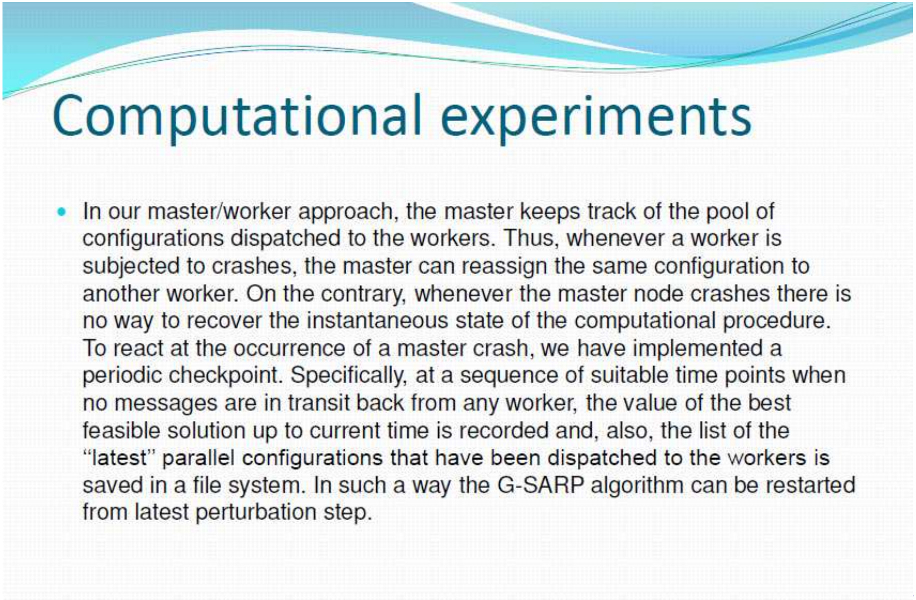
while $T<T_f$

Receive the configuration

Evaluate the new configuration;
if the solution is accepted then send the configuration to the master and the related objective function value (estimate).

Fig. 5. The worker process.

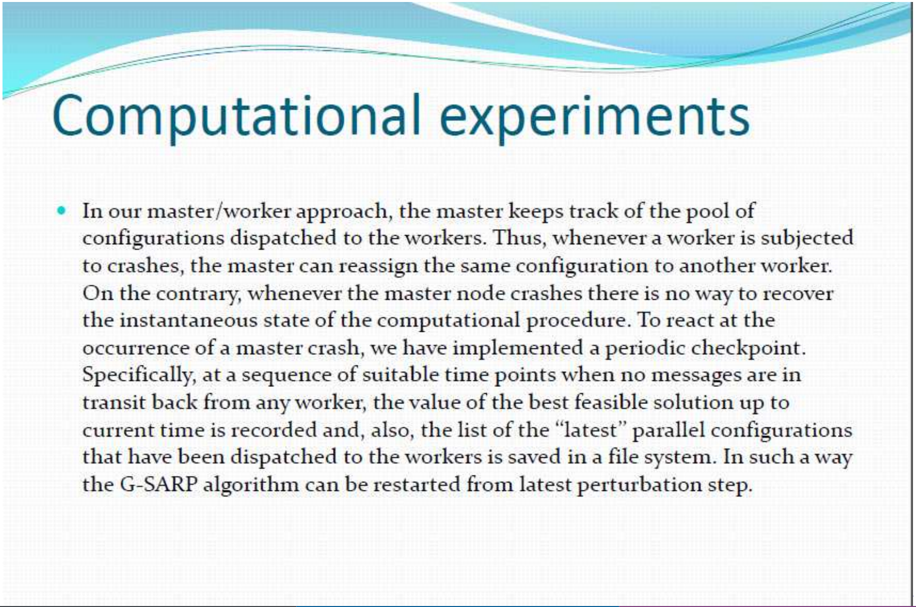
Computational expriments



Computational experiments

- In our master/worker approach, the master keeps track of the pool of configurations dispatched to the workers. Thus, whenever a worker is subjected to crashes, the master can reassign the same configuration to another worker. On the contrary, whenever the master node crashes there is no way to recover the instantaneous state of the computational procedure. To react at the occurrence of a master crash, we have implemented a periodic checkpoint. Specifically, at a sequence of suitable time points when no messages are in transit back from any worker, the value of the best feasible solution up to current time is recorded and, also, the list of the "latest" parallel configurations that have been dispatched to the workers is saved in a file system. In such a way the G-SARP algorithm can be restarted from latest perturbation step.

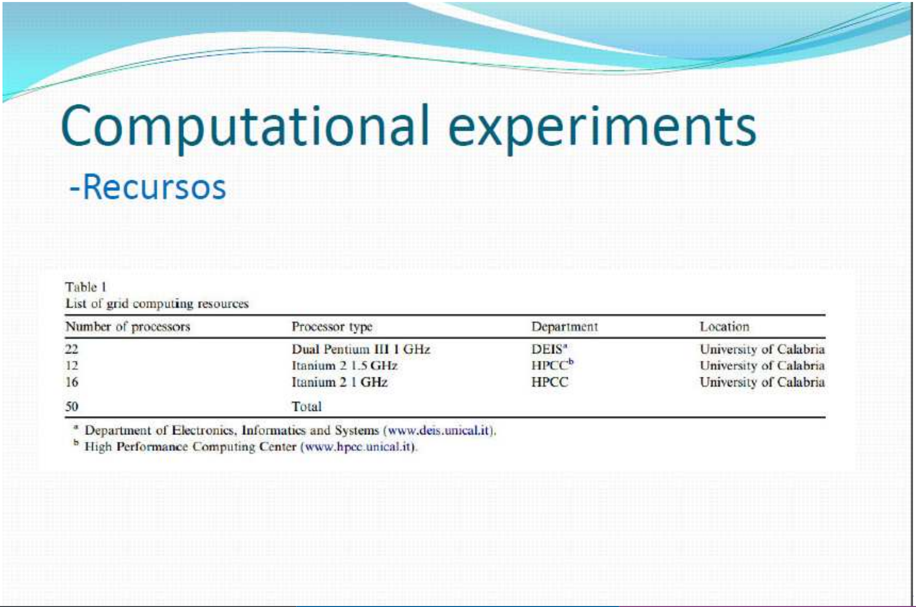
Computational expriments



Computational experiments

- In our master/worker approach, the master keeps track of the pool of configurations dispatched to the workers. Thus, whenever a worker is subjected to crashes, the master can reassign the same configuration to another worker. On the contrary, whenever the master node crashes there is no way to recover the instantaneous state of the computational procedure. To react at the occurrence of a master crash, we have implemented a periodic checkpoint. Specifically, at a sequence of suitable time points when no messages are in transit back from any worker, the value of the best feasible solution up to current time is recorded and, also, the list of the "latest" parallel configurations that have been dispatched to the workers is saved in a file system. In such a way the G-SARP algorithm can be restarted from latest perturbation step.

Recurses



Computational experiments

-Recursos

Table 1
List of grid computing resources

Number of processors	Processor type	Department	Location
22	Dual Pentium III 1 GHz	DEIS[a]	University of Calabria
12	Itanium 2 1.5 GHz	HPCC[b]	University of Calabria
16	Itanium 2 1 GHz	HPCC	University of Calabria
50	Total		

[a] Department of Electronics, Informatics and Systems (www.deis.unical.it).
[b] High Performance Computing Center (www.hpcc.unical.it).

Computational expriments

Computational experiments

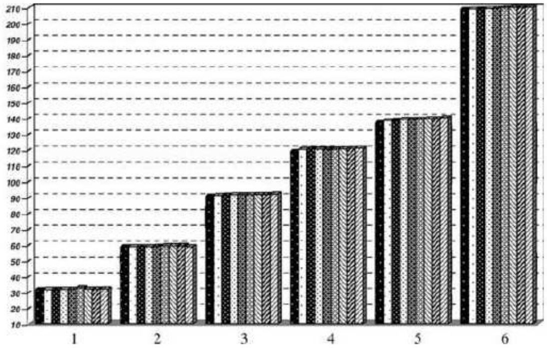
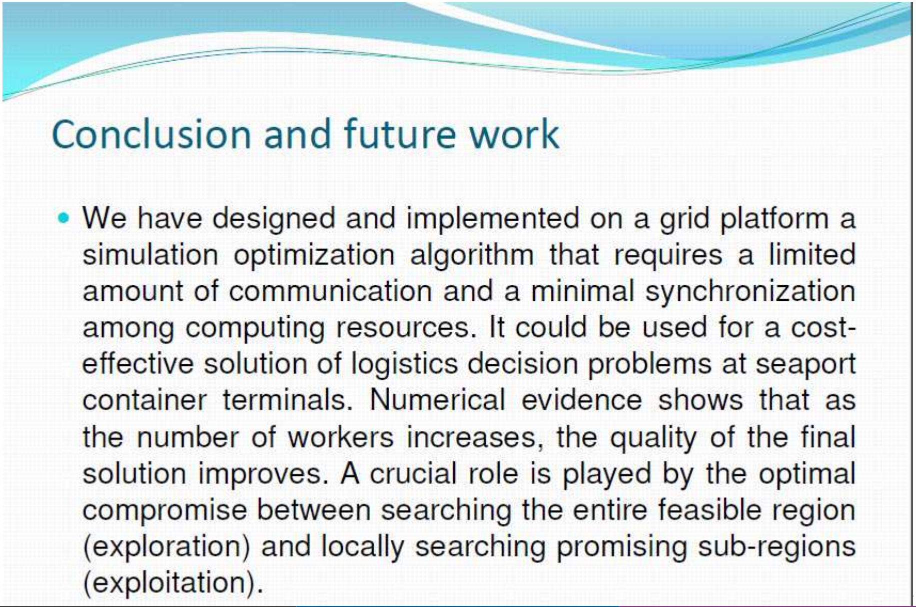


Fig. 6. Solution values for instances 1–6 for increasing number of processors.

Conclucion



Conclusion and future work

- We have designed and implemented on a grid platform a simulation optimization algorithm that requires a limited amount of communication and a minimal synchronization among computing resources. It could be used for a cost-effective solution of logistics decision problems at seaport container terminals. Numerical evidence shows that as the number of workers increases, the quality of the final solution improves. A crucial role is played by the optimal compromise between searching the entire feasible region (exploration) and locally searching promising sub-regions (exploitation).

Gracias por su atencion