Using extremal optimization for Java program initial placement in clusters of JVMs

E. Laskowski[1], M. Tudruj[1,3], I. De Falco[2], U. Scafuri[2], E. Tarantino[2], R. Olejnik[4]

[1]Institute of Computer Science, Polish Academy of Sciences, Warsaw, Poland

[2]Institute of High Performance Computing and Networking, ICAR-CNR, Naples, Italy

[3]Polish-Japanese Institute of Information Technology, Warsaw, Poland

[4]Computer Science Laboratory of Lille, University of Science and Technology of Lille, France.

{laskowsk, tudruj}@ipipan.waw.pl
de.falco@icar.na.it
Richard.Olejnik@lifl.fr

Motivation

- Efficient **load balancing on Grid platform**
- Distribution management:
  - load metrics: CPU queue length, resource utilization, response time
  - communications metrics: transferred data volume, message exchange frequency.
- Balancing strategies:
  - optimization of **initial distribution of components of an application** (initial object deployment)
  - dynamic load balancing (migration of objects).

Initial deployment optimization steps

- Measure the properties of the environment (CPU power and availability, network utilization).
- Execute a program for some representative data (data sample):
  - carry out the measurements of the number of mutual method calls and data volume
  - create a method call graph (a DAG) with the use of method dependency graph and measured data.
- Find the optimal mapping of the graph
- Deploy and run the application in ProActive, a Java-based framework for cluster and Grid computing

EO algorithm

- An introductory optimization algorithm determines an initial distribution of application components on JVMs located on Grid nodes
- The problem is to assign each subtask to one node in the grid in a way that the execution of the application task is as efficient as possible
  - the optimal mapping of application tasks onto the nodes in heterogeneous environment is NP–hard
- So, we use the **Extremal Optimization algorithm** for mapping of tasks to nodes

The principle of the EO

- Extremal Optimization is a **co-evolutionary algorithm** proposed by Boettcher and Percus in 1999
- EO works with **one single solution S** made of a given number of components s_i, each of which is a variable of the problem, is thought to be a species of the ecosystem, and is assigned a fitness value φ_i
- Two fitness functions, one for the variables and one for the global solution.

The outline of the EO

- an initial random solution S is generated and its fitness $\Phi(S)$ is computed

- repeat the following until a termination criterion becomes satisfied:
  - the fitness value φ_i is computed for each of the components s_i
  - the worst variable (in terms of φ_i) is randomly updated, so that the solution is transformed into another solution S' belonging to its neighborhood Neigh(S)

Pseudocode of the τ-EO algorithm

```
begin
  initialize configuration S at will
  set S_best := S
  while maximum number of iterations N_iter not reached do
    evaluate φ_i for each variable s_i of the current solution S
    rank the variables s_i based on their fitness φ_i
    choose the rank k according to k^{-τ} so that the variable s_j with j = π(k) is selected
    choose S' ∈ Neigh(S) such that s_j must change
    accept S := S' unconditionally
    if Φ(S) < Φ(S_best) then
      set S_best := S
    end if
  end while
  return S_best and Φ(S_best)
end
```

The only algorithm parameters are:

- the maximum number of iterations N_{iter}
- the probabilistic selection value τ

Fitness of a solution

- Fitness function of a mapping solution:

$$\Phi(\boldsymbol{\mu}) = \max_{i \in [1,P]} \{\phi_i\}$$

where

$$\phi_i \equiv \phi(i, \mu_i) = \theta_{ij}$$

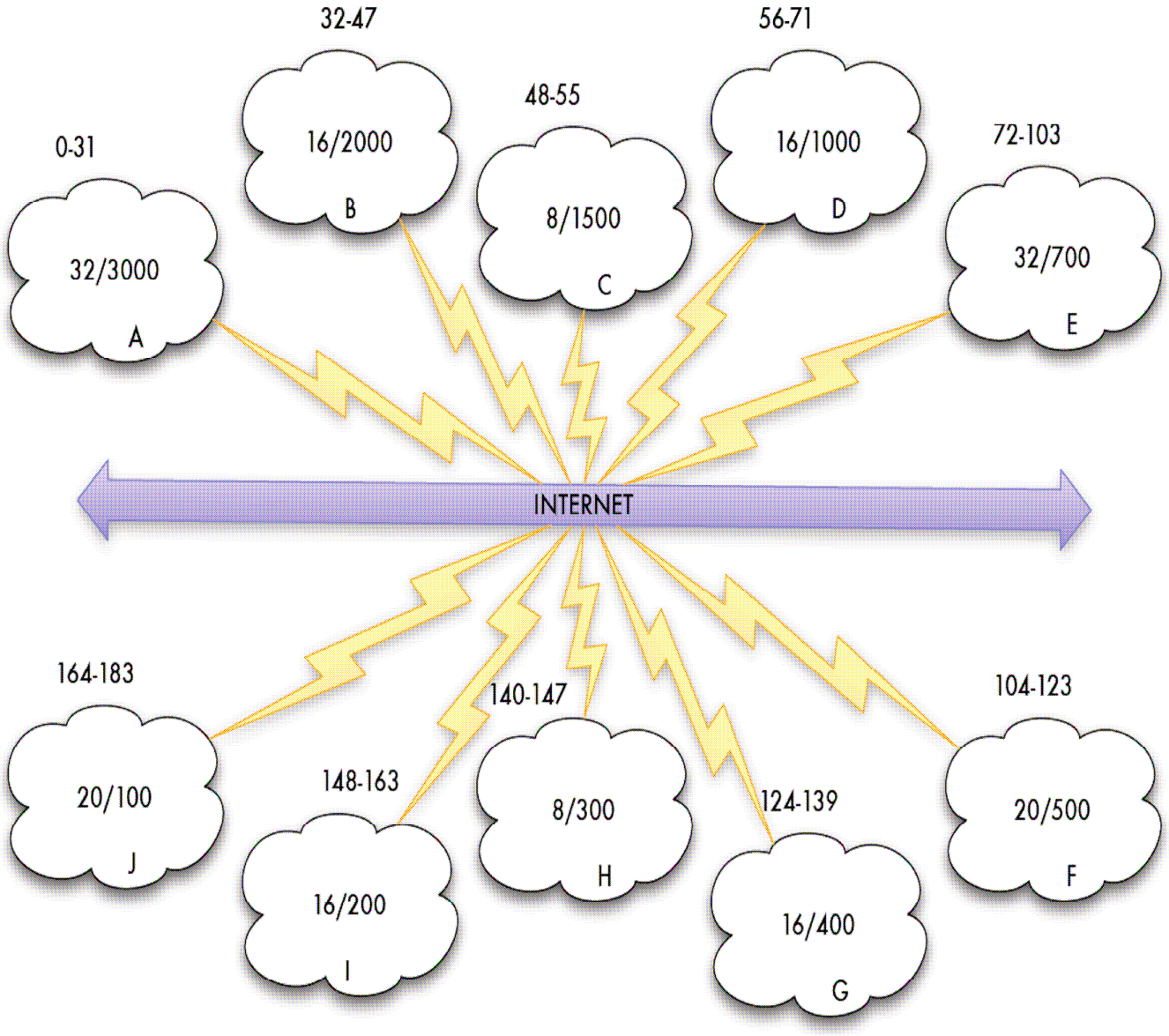
  - θ_{ij}^{comp} the computation time needed to execute the subtask *i* on the node *j* to which it is assigned by the proposed mapping solution
  - θ_{ij}^{comm} the communication time requested to execute the subtask *i* on the node *j* to which it is assigned by the proposed mapping solution
  - $\theta_{ij} = \theta_{ij}^{comp} + \theta_{ij}^{comm}$ is the total time needed to execute the subtask *i* on the node *j* to which it is assigned by the proposed mapping solution.

Experimental results

- τ-EO parameter setting
  - N_{iter} = 200,000
  - τ = 3.0
  - 20 runs on each problem
- Two experiments reported in the presentation:
  - a simulated execution of an application in a test grid (10 sites, 184 nodes with different power and load)
  - an optimization of a ProActive application in a cluster (7 homogenous, two-core nodes).

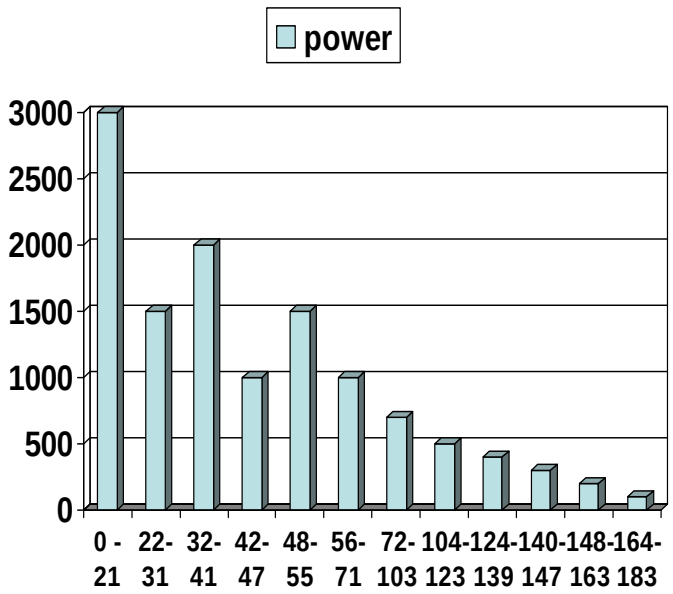
Experiment 1 – the test Grid

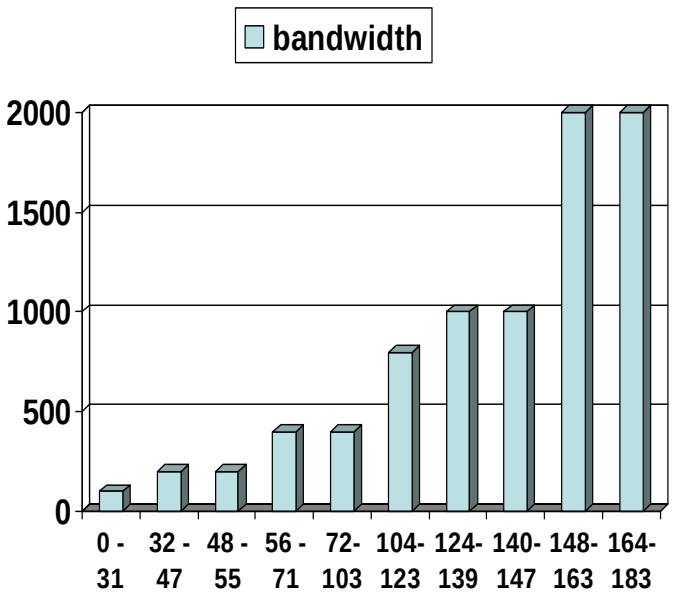
- A Grid with 10 sites, a total of 184 nodes



Experiment 1 - features of the nodes

- Average loads: $\ell_i(\Delta t) = 0.0$ for all nodes apart:
  - $\ell_i(\Delta t) = 0.5$ for each i ϵ [22, …, 31]
  - $\ell_i(\Delta t) = 0.5$ for each i ϵ [42, …, 47]
    - the most powerful nodes are the first 22 of A and the first 10 of B





Experiment 1 – the application

A mapping solution is represented by a vector μ of P integers ranging in the interval [1,N]

12	7	5	4	…	24	9	18
Sub-task 1	Sub-task 2	Sub-task 3	Sub-task 4	…	Sub-task P-2	Sub-task P-1	Sub-task P

In this example the first subtask of the task is placed on the grid node denoted with the number 12, the second on grid node 7, and so on.

Experiment 1 – the results

- The optimal allocation entails both the use of the most powerful nodes and the distribution of the communicating tasks in pairs on the same site so that communications are faster (only intersite, no intrasite)
- the solution allocates 11 task pairs on the 22 unloaded nodes in **A** and the remaining 4 pairs on 8 unloaded nodes in **B**:

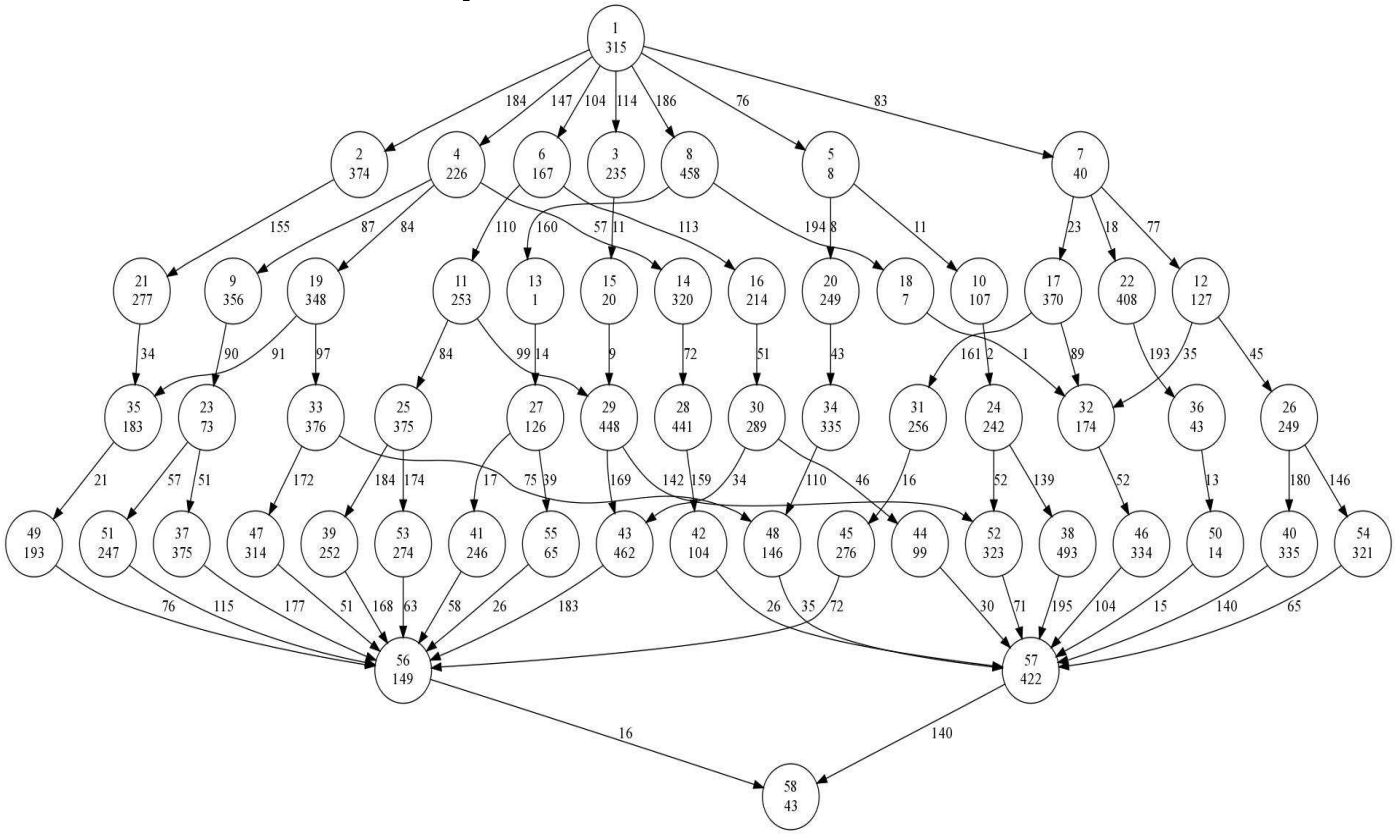
2	22	41	12	20	17	21	13	35	16	18	4	14	39	40
23	1	37	9	11	3	8	10	38	19	5	15	6	33	32

Experiment 2 – the basics

- The cluster:
  - 7 homogenous two-core nodes,
  - Gigabit Ethernet LAN,
  - average extra load $\ell_i(\Delta t) = 0.0$ for all nodes,
  - each node has Sun JVM installed and a ssh agent.
- The scenario of the experiment:
  1. CPU power, load and network utilization monitoring,
  2. application parameters' measuring (using the sample data),
  3. mapping optimization and the final run.

Experiment 2 – the application

- A ProActive Java multi-threaded application, working according to the DAG model
  - 58 nodes
  - the DAG is executed in the loop (200 iterations)



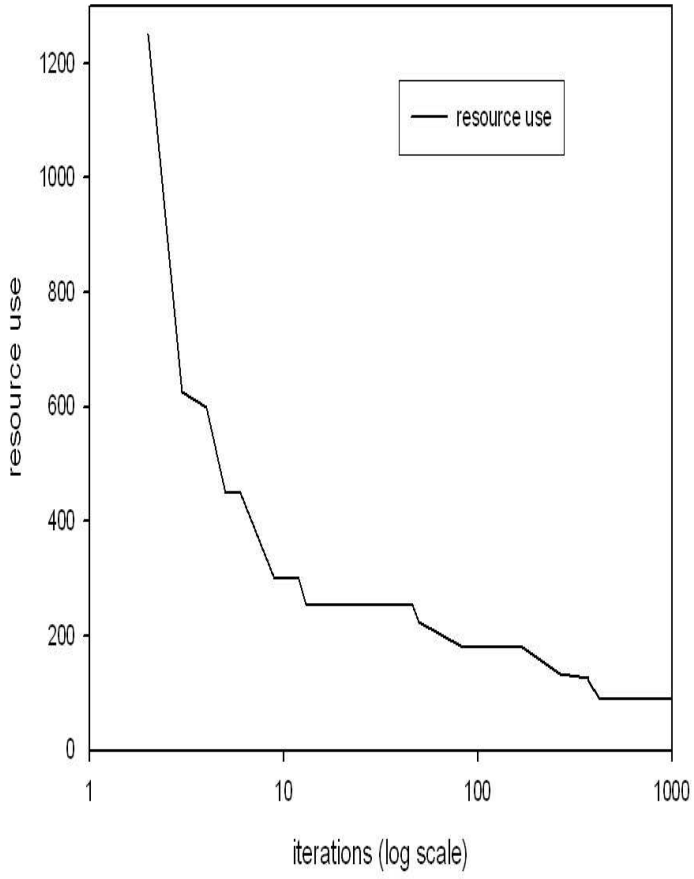
Experiment 2 – the results

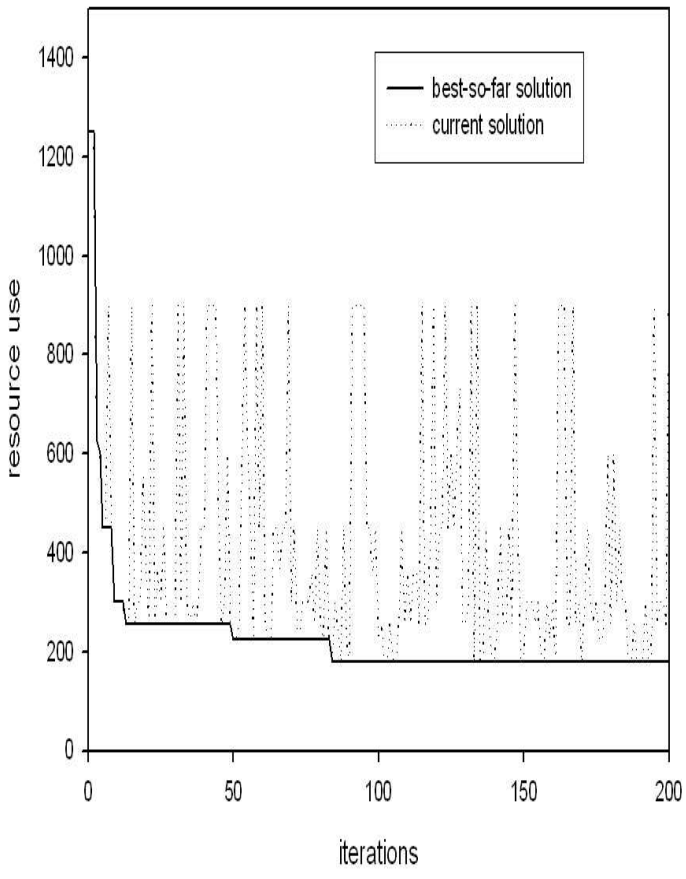
- Since the nodes are homogenous and without the extra load, the EO mapping balanced the amount of computations assigned to each node:

Node nb:	Amount of computations
0	1999
1	2005
2	1993
3	2004
4	2002
5	1980
6	1994

Typical evolution of τ-EO on a mapping problem

- Evolution of the best-so-far value is shown on the left, and both best-so-far and current solutions for the first 200 iterations are shown on the right





Conclusions

- Extremal Optimization has been proposed as a viable approach to the mapping of the tasks making up an application in grid environments
- The unique feature of the presented approach is the ability to deal with different load of nodes and the diversity in network bandwidths
- τ-EO shows two very interesting features when compared to other optimization tools based on Evolutionary Algorithms (e.g. Differential Evolution:
  - a much higher speed
  - its ability to provide stable solutions.