Improving Metaheuristics for Mapping Independent Tasks into Heterogeneous Memory-Constrained System

Javier Cuenca
Domingo Giménez

University of Murcia
SPAIN

Introduction

- Mapping independent tasks to the processors in a heterogeneous system
- Master-slave scheme :
  - The tasks are generated by a processor and sent to other processors which solve them and return the solutions to the initial one
- In our approach:
  - Each task:
    - a computational cost
    - a memory requirement
  - Each processor:
    - a speeds
    - a certain amount of memory → restriction on the tasks can be assigned
- The goal is to obtain a task mapping which leads to a low total execution time.
- The general case is an NP problem → heuristic methods preferable

Scheduling Problem

- The problem:
  - fixed arithmetic costs
  - no communications
  - t tasks:
    - arithmetic costs $c = (c_0, c_1, \ldots, c_{t-1})$
    - memory requirements $i = (i_0, i_1, \ldots, i_{t-1})$
  - p processors
    - the times to perform a basic arithmetic operation $a = (a_0, a_1, \ldots, a_{p-1})$,
    - memory capacities $m = (m_0, m_1, \ldots, m_{p-1})$,

Scheduling Problem

- The problem:
  - from all the mappings, $d = (d_{0,} d_1, \ldots, d_{t-1})$ ($d_k = j$ means task k is assigned to processor j), with $i_k \leq m_{dk}$, find d with which the following mimimum is obtained:

$$\min_{\{d/\ i_k \leq m_{d_k} \forall k=0,1,\ldots,t-1\}} \max_{\{j=0,1,\ldots,p-1\}} \left\{ a_j \sum_{l=0,1,\ldots,t-1;d_l=j} c_l \right\}$$

  - A maximum of p^t assignations → not possible to solve the problem with a reasonable time by generating all the possible mappings
  - An alternative: an approximate solution using some heuristic method

Application of Metaheuristics to the Scheduling Problem

- Application of metaheuristic methods to the version of the scheduling problem previously described
- The methods considered
  - Genetic Algorithm (GA)
  - Scatter Search (SS)
  - Tabu Search (TS)
  - GRASP (GR)
- The goal:
  - to obtain a mapping with:
    - an associated modelled time close to the optimum
    - a low assignation time

Application of Metaheuristics to the Scheduling Problem

Algorithm 1: General scheme of a metaheuristic method.

```
Initialize(S);
while not EndCondition(S) do
    SS = ObtainSubset(S);
    if |SS| > 1 then
        SS1 = Combine(SS);
    else
        SS1 = SS;
    end
    SS2 = Improve(SS1);
    S = IncludeSolutions(SS2);
end
```

Initialize. To create each individual of the initial set *S*. Assigns tasks to processors with a probability proportional to the processor speed

- **GA**: a large initial population of assignations
- **SS**: a reduced number of elements in *S*
- **TS:** a set *S* with only one element.
- **GR**: In each iteration:
    - the cost of each candidate is evaluated
    - a number of candidates are selected to be included in the set of solutions.

ObtainSubset: Some of the individuals are selected randomly.

- **GA**: The individuals with better fitness function have more likelihood of being selected.
- **SS**: It is possible to select all the elements for combination, or to select the best elements to be combined with the worst ones.
- **TS**: This function is not necessary because |*S*| = 1.
- **GR**: One element from the set of solutions is selected to constitute the set *SS* (|*SS*| = 1).

Combine: The selected individuals are crossed, and *SS*1 is obtained.

- **GA**, **SS**: The individuals can be crossed in different ways.
- **TS**, **GR**: This function is not necessary.

Improve:

- **GA**: A few individuals are selected to obtain other individuals, which can differ greatly (mutation operands).
- **SS**: A greedy method. Evaluating the fitness value of the elements obtained with the *p* possible processors in each component.
- **TS**: Some elements in the neighborhood are analysed, excluding tabu elements.
- **GR**: This function consists of a local search to improve the element selected.

IncludeSolutions: Selects some elements of *SS*2 to be included in *S* for the next iteration.

GA: The best individuals from the original set, their descendants and the individuals obtained by mutation.

SS: The best elements are selected, as well as some elements scattered → to avoid falling within local minimums.

TS, **GR**: The best element from those analysed is taken as the next solution.

EndCondition:

GA, **SS**, **TS**, **GR**: maximum number of iterations, or that the best fittness value does not change over a number of iterations.

Application of Metaheuristics to the Scheduling Problem:
Basic Experimental Tuning of the Metaheuristics

- Experiments with different tasks and systems configurations have been carried out, obtaining similar results.
- The experiments have the following configuration:
  - Each Task:
    - The size randomly generated between 1000 and 2000
    - The arithmetic cost is n^3
    - The memory requirement n^2
  - The number of processors in the system is the same as the number of tasks.
  - The costs of basic arithmetic operations: randomly generated between 0.1 and 0.2 *μsecs*.
  - The memory of each processor is between half the memory needed by the biggest task and one and a half times this memory.

Application of Metaheuristics to the Scheduling Problem:
Basic Experimental Tuning of the Metaheuristics

Comparison of backtracking and the metaheuristics. Mapping time and modelled execution time (in seconds), varying the number of tasks.

	Back		GA		SS		TS		GR	
tasks	map.	simul.	map.	simul.	map.	simul.	map.	simul.	map.	simul.
4	0.025	3132	0.051	3132	0.065	3132	0.010	3132	0.019	3132
8	0.034	4731	0.028	4731	0.132	4731	0.015	4731	0.024	4731
12	0.058	1923	0.021	1923	0.158	1923	0.016	**2256**	0.029	1923
13	0.132	1278	0.055	1278	0.159	1278	0.016	**1376**	0.024	1278
14	0.791	1124	0.081	1124	0.192	1124	0.017	1124	0.027	**1135**

Application of Metaheuristics to the Scheduling Problem:

Basic Experimental Tuning of the Metaheuristics

Comparison of the metaheuristics for big systems. Mapping time and modelled execution time (in seconds), varying the number of tasks

	GA		SS		TS		GR	
tasks	map.	simul.	map.	simul.	map.	simul.	map.	simul.
25	0.139	1484	0.259	**1450**	0.010	**1450**	0.045	**1450**
50	0.413	1566	0.429	1900	0.015	1757	0.078	**1524**
100	0.592	1903	0.834	1961	0.022	3018	0.158	**1460**
200	0.825	**3452**	1.540	**3452**	0.079	**3452**	0.293	**3452**
400	3.203	**3069**	2.682	3910	0.375	**3069**	0.698	**3069**

Application of Metaheuristics to the Scheduling Problem:
Advance Tuning of the Genetic Algorithm

- In **Combine**: to change the heredity method:
  - **T1**: Each component is inherited pseudo-randomly, giving more probability to the parent with best fittness value.
  - **T2**. choosing each component of a descendant from the less loaded processor from those of its parents.
    - The load of a processor *r*, W_r:

$$W_r = a_r \sum_{\{l=0,1,\ldots,t-1;d_l=r\}} c_l$$

Application of Metaheuristics to the Scheduling Problem:

Advance Tuning of the Genetic Algorithm

- **T3**. In **Improve:** a hybrid approach, using a steered mutation:
  - Each task assigned to an overloaded processor is reassigned randomly to another processor.
  - →The solution mutates to another where the total loads of the most overload processors have been reduced.

- **T4**. In **ObtainSubset:**
  - To chose pseudo-randomly the solutions that will be combined, giving more probability to the solutions with better fittness.

Application of Metaheuristics to the Scheduling Problem:

Advance Tuning of the Genetic Algorithm

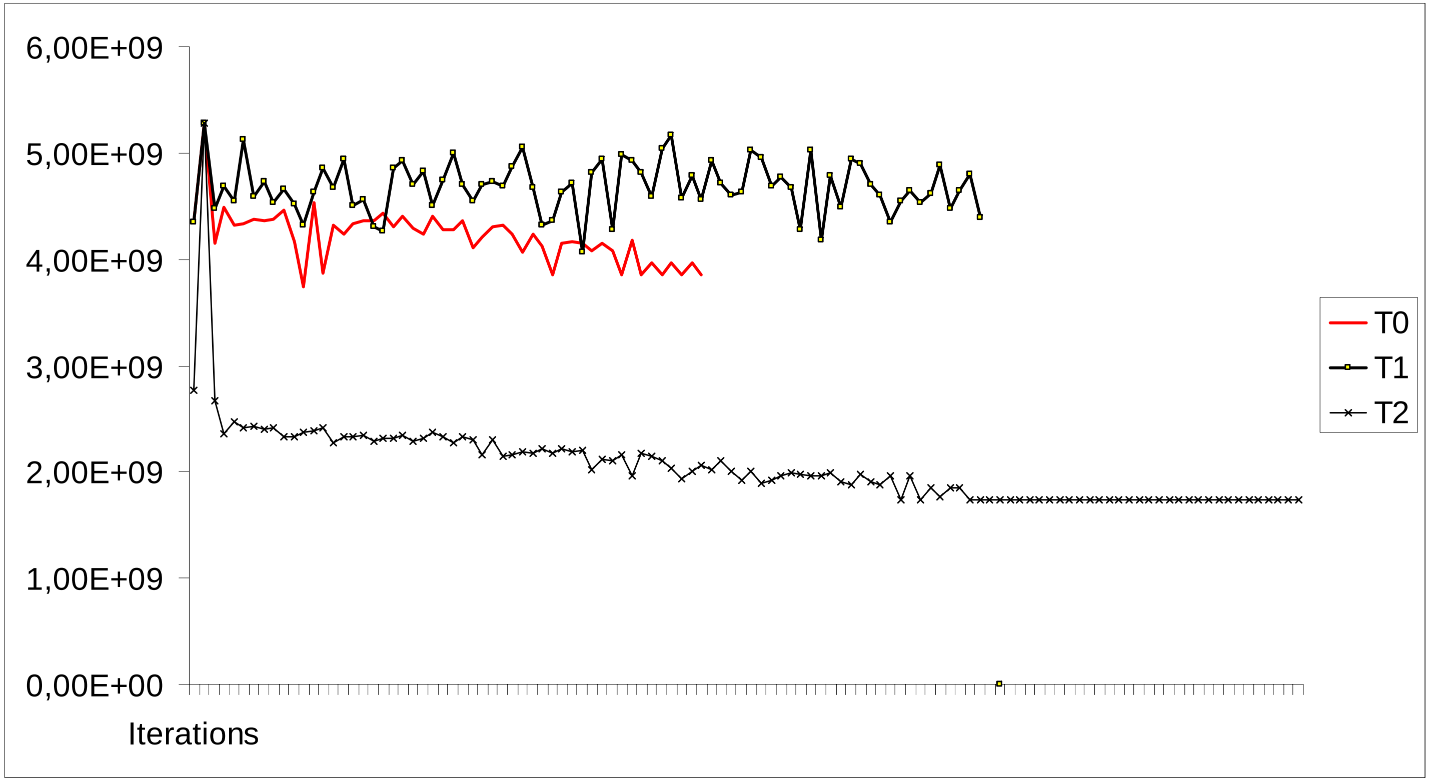
Comparison of the different tunings applied to the Genetic Algorithm, varying the number of tasks

	basic GA		T1		T2		T3		T4		T2+T3		T2+T4		T3+T4	
tasks	map.	simul.	map.	simul.	map.	simul.	map.	simul.	map.	simul.	map.	simul.	map.	simul.	map.	simul.
50	0.13	1646	0.02	2277	0.05	**1524**	0.08	1715	0.09	1715	0.05	**1524**	0.06	**1524**	0.08	1715
100	0.25	2068	0.09	2581	0.13	**1460**	0.14	2230	0.25	2000	0.17	**1460**	0.16	**1460**	0.14	2230
150	0.47	2422	0.19	2908	0.19	**2039**	0.25	2464	0.36	2418	0.22	**2039**	0.22	**2039**	0.25	2464
200	0.41	**3452**	0.28	3717	0.31	**3452**	0.31	**3452**	0.33	**3452**	0.34	**3452**	0.34	**3452**	0.33	**3452**
400	1.56	**3069**	1.19	4184	1.19	**3069**	1.67	**3069**	1.42	**3069**	1.20	**3069**	1.25	**3069**	1.72	**3069**
1600	12.10	3680	10.50	4061	11.77	**1735**	11.38	3882	12.08	3482	12.56	**1735**	11.28	**1735**	12.09	3882

Application of Metaheuristics to the Scheduling Problem:

Advance Tuning of the Genetic Algorithm

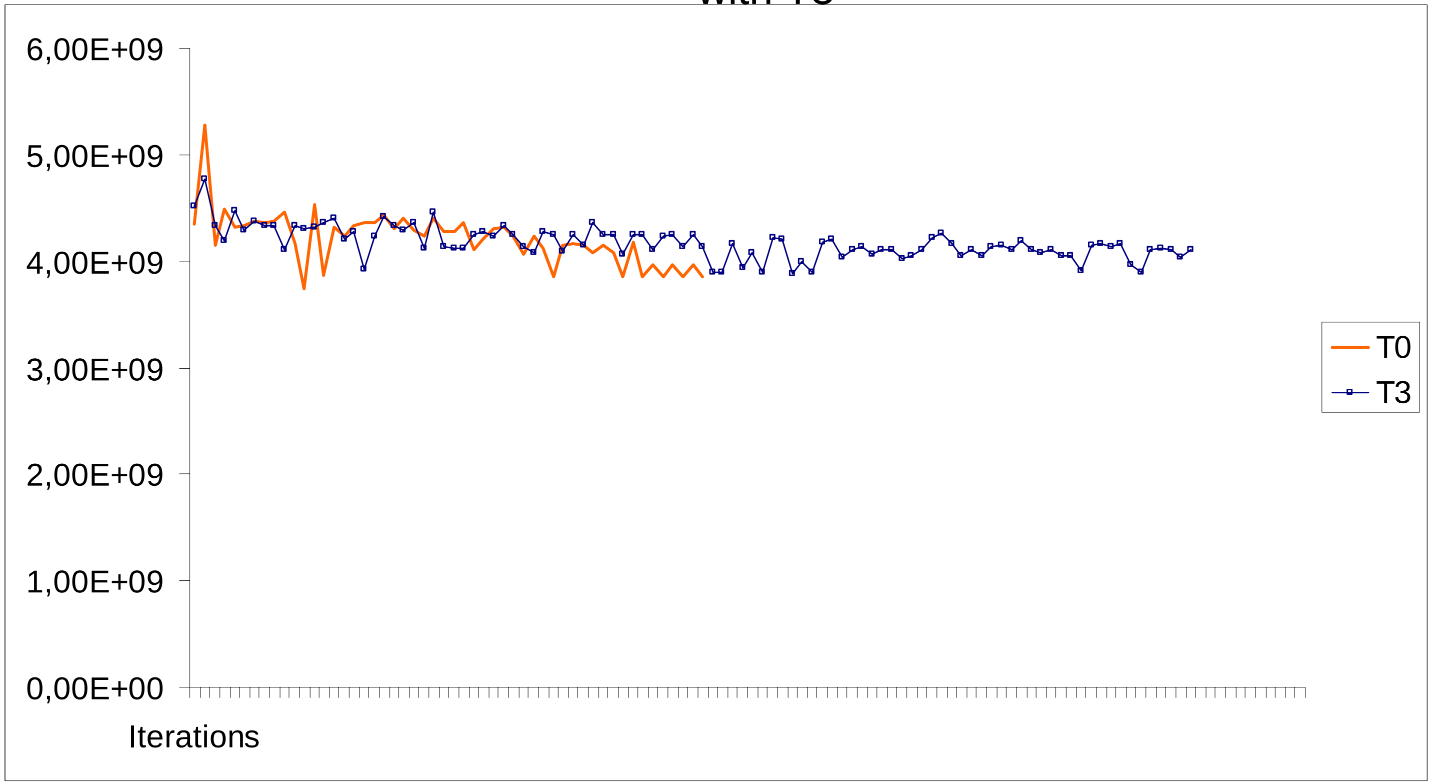
Evolution of the best solution from the new generated individuals per iteration for a problem size of 1600 tasks. Without tuning (T0) applied to the routine **Combine**, with T1 and with T2



Application of Metaheuristics to the Scheduling Problem:

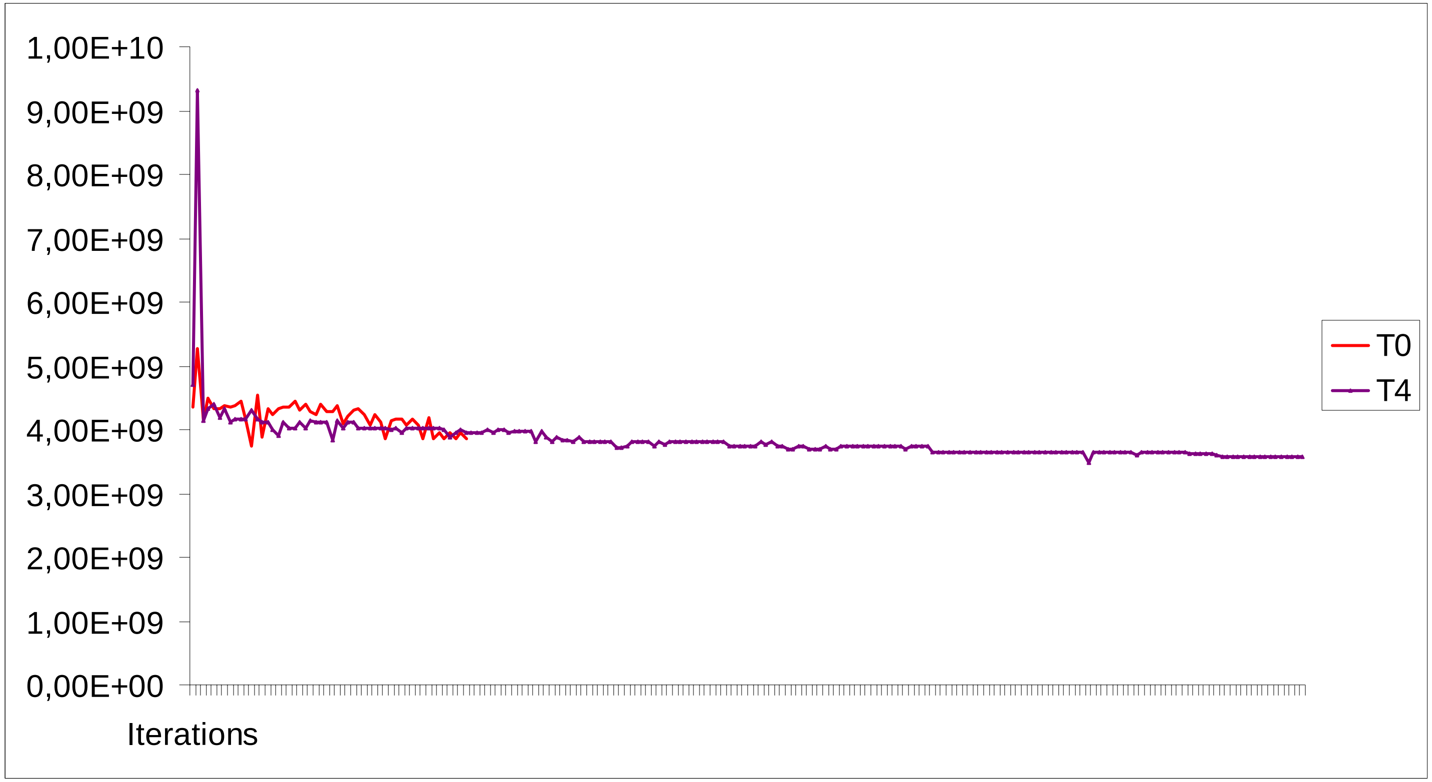
Advance Tuning of the Genetic Algorithm

Evolution of the best solution from the new generated individuals per iteration for a problem size of 1600 tasks. Without tuning (T0) applied to the routine **Improve**, and with T3



Application of Metaheuristics to the Scheduling Problem:
Advance Tuning of the Genetic Algorithm

Evolution of the best solution from the new generated individuals per iteration for a problem size of 1600 tasks. Without tuning (T0) applied to the routine **ObtainSubset**, and with T4



Conclusions and Future Works

- Some improvements of metaheuristics techniques to tasks to processors mapping problems:
  - The tasks
    - Independent
    - Various computational costs and memory requirements
  - The computational system:
    - Heterogeneous
    - Different memory capacities (communications are not yet considered).

- The experiments to obtain satisfactory versions of the metaheuristics have been carried out
  - mainly with the **GA** where some detailed tuning techniques have been studied.

- Future works
  - Advanced tunings, like those applied to the **GA** in this work, will be applied to the other metaheuristics
  - Different characteristics of the heterogeneous systems:
    - variable arithmetic cost in each processor depending on the problem size
    - variable communication cost in each link,...
  - Other general approximations (dynamic assignation of tasks, adaptive metaheuristics,...)