Application of metaheuristics to tasks-to-processors assignation problems in heterogeneous systems

Francisco Almeida
Universidad de La Laguna

Javier Cuenca, Domingo Giménez
Universidad de Murcia

Juan Pedro Martínez Gallar
Universidad Miguel Hernández de Elche
Outline

• The problem
• Mapping trees
• Metaheuristic scheme
• A unified class hierarchy for the application of metaheuristics
• A case study
• Conclusions and Future Works
The problem

- Mapping problems. For example:
  - Master-slave scheme
  - Homogeneous processes
- Theoretical model of the execution time.
- A tree of possible mappings. Each node has associated the theoretical cost of the mapping.
- Optimization problem: To obtain a mapping which leads to the lowest total execution time.
- The general case is an NP problem → heuristic methods compulsory.
- A large number of metaheuristics → facilitate development, reuse and experimentation.
Mapping tree I

Tree of processes to processors mappings, each node has an estimated time. Obtain the node with lowest time.
Mapping tree I: Execution time model

- Heterogeneous computation: $t_{ci}$
- Heterogeneous communication: $t_{sij}$, $t_{wij}$
- Execution time:

$$t = t_c \cdot t_{comp} + t_s \cdot t_{start} + t_w \cdot t_{word}$$

$$t_c = \max\{d_i \cdot t_{ci}\}$$

$$t_s = \max\{d_i \neq 0, d_j \neq 0 : t_{sij}\}$$

$$t_w = \max\{d_i \neq 0, d_j \neq 0 : t_{wij}\}$$
Mapping tree I: Execution time model

\[ t_c = \max\{t_{c1}, 2t_{c2}\} \]
\[ t_s = \max\{t_{s12}, t_{s21}, t_{s22}\} \]
\[ t_w = \max\{t_{w12}, t_{w21}, t_{w22}\} \]
Mapping tree II

A set of tasks generated by a master processor, with memory constraints

nodes eliminated due to memory constraints

Task 1

Task 2

Task 3
Mapping tree II: Execution time model

- Heterogeneous computation: $t_{ci}$
- No communications
- Execution time of each node (mapping $d$):
  $$\max_{j=1,\ldots,P} \left\{ t_{cj} \sum_{l=1,\ldots,T, d_l=j} c_l \right\}$$
- Optimization problem:
  $$\min_d \max_{j=1,\ldots,P} \left\{ t_{cj} \sum_{l=1,\ldots,T, d_l=j} c_l \right\}$$
Metaheuristic scheme

Initialize(C)
WHILE (NOT EndCondition(C))
    S = ObtainSubset(C)
    IF |S|>1
        S1 = Combine(S)
    ELSE
        S1 = S
    ENDIF
    S2 = Improve(S1)
    C = IncludeSolutions(S2)
ENDWHILE

Substitute routines for the particular metaheuristic →
class hierarchy
Metaheuristic scheme

Initialize(C)

WHILE (NOT EndCondition(C))

S = ObtainSubset(C)

IF |S|>1

S1 = Combine(S)

ELSE

S1 = S

ENDIF

S2 = Improve(S1)

C = IncludeSolutions(S2)

ENDWHILE

Initialize. To create each individual of the initial set S. Assign 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**: a set is initially generated, or one element could be generated in each iteration
Metaheuristic scheme

Initialize(C)

WHILE (NOT EndCondition(C))

\[ S = \text{ObtainSubset}(C) \]

IF \(|S|>1\)

\[ S1 = \text{Combine}(S) \]

ELSE

\[ S1 = S \]

ENDIF

\[ S2 = \text{Improve}(S1) \]

\[ C = \text{IncludeSolutions}(S2) \]

ENDWHILE

\textbf{ObtainSubset}: Some of the individuals are selected randomly.

- \textbf{GA}: Pairs of individuals are selected.
- \textbf{SS}: It is possible to select all the elements for combination, or to select the best elements to be combined with the worst ones.
- \textbf{TS}: This function is not necessary because \(|S| = 1\).
- \textbf{GR}: One element from the set of solutions is selected to constitute the set \(SS (|SS| = 1)\). More probability is assigned to mappings with low cost.
Metaheuristic scheme

Initialize(C)
WHILE (NOT EndCondition(C))
    S = ObtainSubset(C)
    IF |S| > 1
        S1 = Combine(S)
    ELSE
        S1 = S
    ENDIF
    S2 = Improve(S1)
    C = IncludeSolutions(S2)
ENDWHILE

**Combine**: The selected individuals are crossed, and SS1 is obtained.

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

Initialize(C)
WHILE (NOT EndCondition(C))
    S = ObtainSubSet(C)
    IF |S| > 1
        S1 = Combine(S)
    ELSE
        S1 = S
    ENDIF
    S2 = Improve(S1)
    C = IncludeSolutions(S2)
ENDWHILE

Improve:
• **GA**: A few individuals are selected to obtain other individuals, which can differ greatly (mutation operands).
• **SS**: A greedy method:
  • Take a task assigned to the most loaded processor.
  • Reassign it to another processor.
  • Repeat while the cost continues to be reduced.
• **TS**: Some elements in the neighborhood are analysed, excluding tabu elements.
• **GR**: The greedy method of **SS** is applied.
Metaheuristic scheme

Initialize(C)
WHILE (NOT EndCondition(C))
    S = ObtainSubset(C)
    IF |S|>1
        S1 = Combine(S)
    ELSE
        S1 = S
    ENDIF
    S2 = Improve(S1)
    C = IncludeSolutions(S2)
ENDWHILE

IncludeSolutions: Selects some elements of SS2 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 scattered elements, to avoid falling within local minima.

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

Initialize(C)

WHILE (NOT EndCondition(C))
    S = ObtainSubset(C)
    IF |S|>1
        S1 = Combine(S)
    ELSE
        S1 = S
    ENDIF
    S2 = Improve(S1)
    C = IncludeSolutions(S2)
ENDWHILE

EndCondition:

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

- Reuse classes and methods
- Develop new metaheuristics
- Tune parameters and functions for the problem
- Obtain a satisfactory metaheuristic for the problem
A case study: problem II

Mapping time and modelled execution time (in seconds), varying the number of tasks

<table>
<thead>
<tr>
<th>tasks</th>
<th>GA map. simul.</th>
<th>SS map. simul.</th>
<th>TS map. simul.</th>
<th>GR map. simul.</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>0.139</td>
<td>0.259</td>
<td>0.010</td>
<td>0.045</td>
</tr>
<tr>
<td>50</td>
<td>0.413</td>
<td>0.429</td>
<td>0.015</td>
<td>0.078</td>
</tr>
<tr>
<td>100</td>
<td>0.592</td>
<td>0.834</td>
<td>0.022</td>
<td>0.158</td>
</tr>
<tr>
<td>200</td>
<td>0.825</td>
<td>1.540</td>
<td>0.079</td>
<td>0.293</td>
</tr>
<tr>
<td>400</td>
<td>3.203</td>
<td>2.682</td>
<td>0.375</td>
<td>0.698</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>1484</th>
<th>1450</th>
<th>1450</th>
<th>1450</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1566</td>
<td>1900</td>
<td>1757</td>
<td>1524</td>
</tr>
<tr>
<td></td>
<td>1903</td>
<td>1961</td>
<td>3018</td>
<td>1460</td>
</tr>
<tr>
<td></td>
<td>3452</td>
<td>3452</td>
<td>3452</td>
<td>3452</td>
</tr>
<tr>
<td></td>
<td>3069</td>
<td>3910</td>
<td>3069</td>
<td>3069</td>
</tr>
</tbody>
</table>
A case study: problem II

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

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

- Hierarchy of metaheuristics
- Easy development, tuning and experimentation

- Application to other mapping problems
- Class hierarchy for general mapping problems
- Reuse among different problems
Thanks

Questions?
Comments?
Criticisms?
Application of Metaheuristics to the Scheduling Problem:

Advanced 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 better fitness value.
  - **T2**: Choosing each component of a descendant from the less loaded processor of those of its parents.
  - The load of a processor $r$:
    \[
    W_r = a_r \sum_{\{l=0,1,\ldots,t-1; d_l = r\}} c_l
    \]
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 randomly reassigned.
  - The goal of this mutation is to reduce the total load of the most overloaded processors.

- **T4. In ObtainSubset:**
  - To chose pseudo-randomly the solutions that will be combined, giving more probability to the solutions with better fitness.
Application of Metaheuristics to the Scheduling Problem:

Advance Tuning of the Genetic Algorithm

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