Motivation

- To tune a metaheuristic to a problem, experiments with several parameters (intra-metaheuristic parameters) and functions.
- To obtain a good metaheuristic for a problem, experiments with several metaheuristics.

We propose the use of unified parallel schemes for metaheuristics:
different values of inter-metaheuristic parameters would provide different metaheuristics or hybridation/combination of metaheuristics.
Motivation

- To tune a metaheuristic to a problem, experiments with several parameters (intra-metaheuristic parameters) and functions
- To obtain a good metaheuristic for a problem, experiments with several metaheuristics

We propose the use of unified parallel schemes for metaheuristics: different values of inter-metaheuristic parameters would provide different metaheuristics or hybridation/combination of metaheuristics.
Motivation

- To tune a metaheuristic to a problem, experiments with several parameters (intra-metaheuristic parameters) and functions
- To obtain a good metaheuristic for a problem, experiments with several metaheuristics

We propose the use of **unified parallel schemes for metaheuristics**: different values of inter-metaheuristic parameters would provide different metaheuristics or hybridation/combination of metaheuristics
Objectives

- To develop unified parameterized schemes of metaheuristics
  Done: combination of GRASP, Genetic algorithm, Scatter search
  Applied to: Simultaneous Equation Models, p-Hub, tasks-to-processors assignment, knapsack 0/1
- From those schemes, develop unified parallel schemes
  Done: on shared-memory, OpenMP
- Future: auto-optimization of the parallel metaheuristics by autonomous selection of the number of threads (processors) to use in each part of the parallel scheme
  Done: parameterization of each function in the unified shared-memory scheme
Objectives

- To develop unified parameterized schemes of metaheuristics
  Done: combination of GRASP, Genetic algorithm, Scatter search
  Applied to: Simultaneous Equation Models, p-Hub, tasks-to-processors assignment, knapsack 0/1

- From those schemes, develop unified parallel schemes
  Done: on shared-memory, OpenMP

- Future: auto-optimization of the parallel metaheuristics by autonomous selection of the number of threads (processors) to use in each part of the parallel scheme
  Done: parameterization of each function in the unified shared-memory scheme
Objectives

- To develop unified parameterized schemes of metaheuristics
  Done: combination of GRASP, Genetic algorithm, Scatter search
  Applied to: Simultaneous Equation Models, p-Hub, tasks-to-processors assignment, knapsack 0/1

- From those schemes, develop unified parallel schemes
  Done: on shared-memory, OpenMP

- Future: auto-optimization of the parallel metaheuristics by autonomous selection of the number of threads (processors) to use in each part of the parallel scheme
  Done: parameterization of each function in the unified shared-memory scheme
 Unified metaheuristic scheme

Inicialize($S$)
while (not EndCondition($S$))
    $SS = \text{Select}(S)$
    $SS1 = \text{Combine}(SS)$
    $SS2 = \text{Improve}(SS1)$
    $S = \text{Include}(SS2)$

Facilitates to work with different metaheuristics by reusing functions
Parameterized metaheuristic scheme

Inicialize($S$,ParamIni)

while (not EndCondition($S$,ParamEnd))
    $SS = Select(S,ParamSel)$
    $SS1 = Combine(SS,ParamCom)$
    $SS2 = Improve(SS1,ParamImp)$
    $S = Include(SS2,ParamInc)$

The use of inter-metaheuristic parameters facilitates to work with different metaheuristics/hybridation/combination by selecting different values of the parameters in the functions.
Unified shared-memory metaheuristics

Identify functions with the same parallel scheme:

Loop parallelism

`omp_set_num_threads(one-loop-threads)`

`#pragma omp parallel for`

`loop in elements`

`treat element`

i.e.: Initialize, Combine...
Unified shared-memory metaheuristics

Nested parallelism

```c
omp_set_num_threads(first-level-threads)
#pragma omp parallel for
loop in elements
treat-element-second-level(first-level-threads)
```

treat-element-second-level(first-level-threads):
  ```c
  omp_set_num_threads(second-level-threads(first-level-threads))
  #pragma omp parallel for
  loop in elements
treat element
  ```
  i.e.: Initialize, Improve...

Allows fine and coarse grained parallelism by changing the number of threads in each level
Metaheuristics

- Pure metaheuristics: GRASP, Genetic algorithms (GA), Scatter search (SS)
- Combinations: GRASP+GA, GRASP+SS, GA+SS, GRASP+GA+SS

![Space of metaheuristics diagram](image-url)
Inicialize

- Randomly generate valid elements. The number of elements ($INEIni$) varies to tune the metaheuristic to the problem (intra parameter), but different values can correspond to different metaheuristics (inter-metaheuristic parameter).
- Generated elements can be improved with local search, greedy..., with a percentage of elements to improve ($PEIIni$) and an intensification in the improvement ($IIEIni$).
- A number of elements ($NERIni$) is selected to form the reference set.

$ParamIni = (INEIni, PEIIni, IIEIni, NERIni)$
Randomly generate valid elements. The number of elements \((INE_{Ini})\) varies to tune the metaheuristic to the problem (intra parameter), but different values can correspond to different metaheuristics (inter-metaheuristic parameter).

Generated elements can be improved with local search, greedy..., with a percentage of elements to improve \((PEI_{Ini})\) and an intensification in the improvement \((IIE_{Ini})\).

A number of elements \((NER_{Ini})\) is selected to form the reference set.

\[
Param_{Ini} = (INE_{Ini}, PEI_{Ini}, IIE_{Ini}, NER_{Ini})
\]
Inicialize

- Randomly generate valid elements. The number of elements ($INE_{Ini}$) varies to tune the metaheuristic to the problem (intra parameter), but different values can correspond to different metaheuristics (inter-metaheuristic parameter).
- Generated elements can be improved with local search, greedy..., with a percentage of elements to improve ($PEI_{Ini}$) and an intensification in the improvement ($IIE_{Ini}$).
- A number of elements ($NER_{Ini}$) is selected to form the reference set.

$$Param_{Ini} = (INE_{Ini}, PEI_{Ini}, IIE_{Ini}, NER_{Ini})$$
Select

- The best and worst elements according to some function (fitness function, scatter function...) are selected.
- The number of best elements is $NBESel$, and the number of worst $NWESel$, and normally $NBESel + NWESel = NERIni$.

$$ParamSel = (NBESel, NWESel)$$
Combine

- A certain number of combinations between the best elements ($NBBCom$),
  between the best and the worst ($NBWCom$),
  and between the worst ($NWWCom$).

$$ParamCom = (NBBCom, NBWCom, NWWCom)$$
A percentage of elements ($PEImp$) are improved by local search..., with a certain intensification ($IIImp$).

A percentage of elements which are “distant” ($PEImp$) to the reference set are generated, and an improvement is applied to these elements with a certain intensification ($IIImp$).

$ParamImp = (PEImp, IIEImp, PEDImp, IIDImp)$
A percentage of elements ($PEIImp$) are improved by local search..., with a certain intensification ($IIEImp$).

A percentage of elements which are “distant” ($PEDImp$) to the reference set are generated, and an improvement is applied to these elements with a certain intensification ($IIDImp$).

$$ParamImp = (PEIImp, IIEImp, PEDImp, IIDImp)$$
Include and EndCondition

- The best $NBE_{inc}$ elements are included in the reference set, and the others ($NER_{ini} - NBE_{inc}$) are the most “distant” to the best ones according to some distance function

$$Param_{inc} = (NNE_{inc})$$

- The method converges when a maximum number of iterations ($MNI_{end}$) or a maximum number of iterations without improving the best solution ($NIRE_{end}$) is performed

$$Param_{end} = (MNI_{end}, NIRE_{end})$$

(they could be consider inter-metaheuristic parameters)
Motivation parameterized metaheuristic scheme Unified shared-memory metaheuristics Experiments Conclusions

Include and EndCondition

- The best $NBE_{Inc}$ elements are included in the reference set, and the others ($NER_{Ini} - NEB_{Inc}$) are the most “distant” to the best ones according to some distance function

\[ \text{Param}_{Inc} = (NNE_{Inc}) \]

- The method converges when a maximum number of iterations ($MNI_{End}$) or a maximum number of iterations without improving the best solution ($NIRE_{End}$) is performed

\[ \text{Param}_{End} = (MNI_{End}, NIRE_{End}) \]

(they could be consider inter-metaheuristic parameters)
Obtaining satisfactory Simultaneous Equation Models from a set of values of the variables:

\[
\begin{align*}
y_1 & = \gamma_{1,1}x_1 + \ldots + \gamma_{1,K}x_K + \beta_{1,2}y_2 + \beta_{1,3}y_3 + \ldots + \beta_{1,N}y_N + u_1 \\
y_2 & = \gamma_{2,1}x_1 + \ldots + \gamma_{2,K}x_K + \beta_{2,1}y_1 + \beta_{2,3}y_3 + \ldots + \beta_{2,N}y_N + u_2 \\
\ldots \\
y_N & = \gamma_{N,1}x_1 + \ldots + \gamma_{N,K}x_K + \beta_{N,1}y_1 + \ldots + \beta_{N,N-1}y_{N-1} + u_N
\end{align*}
\]

given values of \(x_i\) and \(y_i\) (vectors of dimension \(d\)) obtain the system (\(\beta_{i,j}\) and \(\gamma_{i,j}\) non equal to zero) which best represents the variables dependencies, according to some criterium (AIC: Akaike Information Criterion)
Applications in econometrics, medicine...
Applications - tasks-to-processors assignation

- $T$ independent tasks to be assigned to $P$ processors. Each task has certain memory requirements and each processor has a certain amount of memory.
  - The tasks have arithmetic costs $c = (c_1, c_2, \ldots, c_T)$ and memory requirements $r = (r_1, r_2, \ldots, r_T)$.
  - The costs of the basic arithmetic operations in the processors are $t_c = (t_{c_1}, t_{c_2}, \ldots, t_{c_P})$, and the memory capacities are $m = (m_1, m_2, \ldots, m_P)$.
  - From all the mappings of tasks to the processors, $d = (d_1, d_2, \ldots, d_T)$, with $r_i \leq m_{d_i}$, find $d$ that minimizes the modeled parallel execution time:

$$\min \left\{ d/ r_k \leq m_{d_k} \forall k=1,2,\ldots,T \right\} \max_{i=1,\ldots,P} \left\{ t_{c_i} \sum_{j=1}^{T} c_j \right\}$$
Computational systems

- **Nodes of Rosebud (cluster of the Polytechnic University of Valencia):**
  - Rosebud4: Intel Core 2 Quad.
  - Rosebud8: Fujitsu Primergy RXi600 with 4 Dual-Core processors Intel Itanium2.

- **Ben–Arabí (Supercomputing Center of Murcia):**
  - Ben: HP Integrity Superdome SX2000 with 128 cores of the processor Intel Itanium-2 dual-core Montvale.
  - Arabí: cluster of 102 nodes, each one with 8 cores of the processor Intel Xeon Quad-Core L5450 (used one node).
satisfactory values for the “pure” metaheuristics obtained by experimentation

<table>
<thead>
<tr>
<th>Parameters</th>
<th>GRASP</th>
<th>GA</th>
<th>SS</th>
<th>GRASP+GA</th>
<th>GRASP+SS</th>
<th>GA+SS</th>
<th>GRASP+GA+SS</th>
</tr>
</thead>
<tbody>
<tr>
<td>INEIni</td>
<td>200 / 100</td>
<td>500 / 100</td>
<td>100</td>
<td>200 / 100</td>
<td>200 / 100</td>
<td>100</td>
<td>200 / 100</td>
</tr>
<tr>
<td>NERIni</td>
<td>-</td>
<td>500 / 100</td>
<td>20</td>
<td>200 / 100</td>
<td>20</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>PEIIIni</td>
<td>100</td>
<td>0</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>IIEIni</td>
<td>10</td>
<td>-</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>NBESel</td>
<td>-</td>
<td>500 / 100</td>
<td>10</td>
<td>200 / 100</td>
<td>10</td>
<td>25 / 37</td>
<td>25 / 37</td>
</tr>
<tr>
<td>NWESel</td>
<td>-</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>10</td>
<td>25/12</td>
<td>25/12</td>
</tr>
<tr>
<td>NBBCCom</td>
<td>-</td>
<td>250 / 50</td>
<td>90 / 45</td>
<td>100 / 50</td>
<td>90 / 45</td>
<td>90 / 666</td>
<td>90 / 666</td>
</tr>
<tr>
<td>NBWCom</td>
<td>-</td>
<td>-</td>
<td>100</td>
<td>-</td>
<td>100</td>
<td>100 / 444</td>
<td>100 / 444</td>
</tr>
<tr>
<td>NWWCom</td>
<td>-</td>
<td>-</td>
<td>90 / 45</td>
<td>-</td>
<td>90 / 45</td>
<td>90 / 66</td>
<td>90 / 66</td>
</tr>
<tr>
<td>PEIImp</td>
<td>-</td>
<td>0</td>
<td>100</td>
<td>0</td>
<td>100</td>
<td>100 / 50</td>
<td>100 / 50</td>
</tr>
<tr>
<td>IIEImp</td>
<td>-</td>
<td>-</td>
<td>5 / 10</td>
<td>-</td>
<td>5 / 10</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>PEDImp</td>
<td>-</td>
<td>10 / 5</td>
<td>0</td>
<td>10 / 5</td>
<td>0</td>
<td>10 / 50</td>
<td>10 / 50</td>
</tr>
<tr>
<td>IIDImp</td>
<td>-</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>-</td>
<td>5 / 0</td>
<td>5 / 0</td>
</tr>
<tr>
<td>NBEInc</td>
<td>-</td>
<td>500 / 100</td>
<td>10</td>
<td>200 / 100</td>
<td>10</td>
<td>25 / 37</td>
<td>25 / 37</td>
</tr>
</tbody>
</table>
SEM, $N = K = 20$, $d = 100$
SEM, $N = K = 20$, $d = 100$

speed-up with the maximum number of cores, the optimum number of threads for the complete program, and different number of threads in the different functions and levels.
Tasks-to-processors, $T = P = 800$

speed-up with the maximum number of cores, the optimum number of threads for the complete program, and different number of threads in the different functions and levels.
The use of parameterized metaheuristics allows:

- To experiment with different metaheuristics and combinations (inter-parameters) to obtain one satisfactory for a particular problem
- To experiment with different parameters (intra-parameters) to tune the metaheuristic to the problem
- To develop unified parallel schemes, which can be optimized by selecting the parallel parameters (number of threads in the different functions) for the particular metaheuristic and problem
Future research

- Inclusion of more “pure” metaheuristics
- Design of hyperheuristics to automatically select the values of the inter-parameter for a particular problem
- Inclusion of auto-optimization in the parallel scheme, with some engine to autonomously select the number of threads
- Develop unified parallel schemes for other computational systems (message-passing, hybrid, GPU...)