Obtaining Simultaneous Equation Models through a unified shared-memory scheme of metaheuristics

Francisco Almeida Departamento de Estadística, Investigación Operativa y Computación, Universidad de La Laguna
Domingo Giménez Departamento de Informática y Sistemas, Universidad de Murcia
Jose Juan López Espín Centro de Investigación Operativa, Universidad Miguel Hernández

Universidad de La Laguna

Parallel Computing and Optimization - IPDPS, Anchorage, Alaska, May 2011

Contents

1. Motivation
2. Obtaining SEM
3. Parametrized metaheuristics
4. Unified shared-memory metaheuristics
5. Experiments
6. Conclusions

Simultaneous Equation Models

- Simultaneous Equation Models (SEM) have been used in econometrics for years (Keynes model). They are used in medicine, network simulation, study of sociological behavior, etc.
- Traditionally, SEM have been developed by people with a wealth of experience in the particular problem represented by the model.
- Our objective is to develop an algorithm which, given a set of values of the variables, finds a satisfactory SEM.
- The space of the possible solutions is very large and exhaustive search methods are not suitable here.
- Our work is on the application of metaheuristics.

Simultaneous Equation Models

- Simultaneous Equation Models (SEM) have been used in econometrics for years (Keynes model). They are used in medicine, network simulation, study of sociological behavior, etc.
- Traditionally, SEM have been developed by people with a wealth of experience in the particular problem represented by the model.
- Our objective is to develop an algorithm which, given a set of values of the variables, finds a satisfactory SEM.
- The space of the possible solutions is very large and exhaustive search methods are not suitable here.
- Our work is on the application of metaheuristics.

Simultaneous Equation Models

- Simultaneous Equation Models (SEM) have been used in econometrics for years (Keynes model). They are used in medicine, network simulation, study of sociological behavior, etc.
- Traditionally, SEM have been developed by people with a wealth of experience in the particular problem represented by the model.
- Our objective is to develop an algorithm which, given a set of values of the variables, finds a satisfactory SEM.
- The space of the possible solutions is very large and exhaustive search methods are not suitable here.
- Our work is on the application of metaheuristics.

Simultaneous Equation Models

- Simultaneous Equation Models (SEM) have been used in econometrics for years (Keynes model). They are used in medicine, network simulation, study of sociological behavior, etc.
- Traditionally, SEM have been developed by people with a wealth of experience in the particular problem represented by the model.
- Our objective is to develop an algorithm which, given a set of values of the variables, finds a satisfactory SEM.
- The space of the possible solutions is very large and exhaustive search methods are not suitable here.
- Our work is on the application of metaheuristics.

Simultaneous Equation Models

- Simultaneous Equation Models (SEM) have been used in econometrics for years (Keynes model). They are used in medicine, network simulation, study of sociological behavior, etc.
- Traditionally, SEM have been developed by people with a wealth of experience in the particular problem represented by the model.
- Our objective is to develop an algorithm which, given a set of values of the variables, finds a satisfactory SEM.
- The space of the possible solutions is very large and exhaustive search methods are not suitable here.
- Our work is on the application of metaheuristics.

Parametrized metaheuristics

- 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

$$\Longrightarrow$$

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

Parametrized metaheuristics

- 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

$\Longrightarrow$

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

Parametrized metaheuristics

- 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

$$\Longrightarrow$$

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

Parallel-parametrized metaheuristics

- To select a satisfactory metaheuristic and to tune it to the problem requires a lot of experiments
- When applying metaheuristics to obtain satisfactory SEM a large number of systems are solved

$$\Longrightarrow$$

- We propose the use of **unified parallel-parametrized schemes for metaheuristics**:
  the different metaheuristics obtained from the parametrized scheme are parallelized together, with parallel parameters for optimization of the execution time

Parallel-parametrized metaheuristics

- To select a satisfactory metaheuristic and to tune it to the problem requires a lot of experiments
- When applying metaheuristics to obtain satisfactory SEM a large number of systems are solved

$$\Longrightarrow$$

- We propose the use of **unified parallel-parametrized schemes for metaheuristics**:
  the different metaheuristics obtained from the parametrized scheme are parallelized together, with parallel parameters for optimization of the execution time

Parallel-parametrized metaheuristics

- To select a satisfactory metaheuristic and to tune it to the problem requires a lot of experiments
- When applying metaheuristics to obtain satisfactory SEM a large number of systems are solved

$$\Longrightarrow$$

- We propose the use of **unified parallel-parametrized schemes for metaheuristics**:
  the different metaheuristics obtained from the parametrized scheme are parallelized together, with parallel parameters for optimization of the execution time

SEM: a system with N equations, N endogenous variables, K exogenous variables and sample size d is:

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

The problem: given values of x_i and y_i (vectors of dimension d, obtained by experimentation or survey), obtain the system ($\beta_{i,j}$ and $\gamma_{i,j}$ non equal to zero) which best represents the variables' dependencies, according to some criterion

- Methods to solve SEM:
  - Maximum Likelihood
  - Indirect Least Square
  - **Two-Steps Least Square (2SLS)**
  - Three-Steps Least Square
- We use 2SLS:
  - Lower computational cost ($O(NK^2d)$)
  - Can be applied in more cases
- ... but the conclusions about the application of metaheuristics do not depend on the method used.

- Methods to solve SEM:
  - Maximum Likelihood
  - Indirect Least Square
  - **Two-Steps Least Square (2SLS)**
  - Three-Steps Least Square
- We use 2SLS:
  - Lower computational cost ($O(NK^2d)$)
  - Can be applied in more cases
- ... but the conclusions about the application of metaheuristics do not depend on the method used.

- Methods to solve SEM:
  - Maximum Likelihood
  - Indirect Least Square
  - **Two-Steps Least Square (2SLS)**
  - Three-Steps Least Square
- We use 2SLS:
  - Lower computational cost ($O(NK^2d)$)
  - Can be applied in more cases
- ... but the conclusions about the application of metaheuristics do not depend on the method used.

- For each variable it should be decided if it is included (1) or not (0) in the system: number of possible models $2^{N(N+K)}$
- Different criteria can be used to determine the goodness of the model. We use the **Akaike Information Criterion**:

$$d \cdot \ln |\hat{\Sigma}_e| + 2 \sum_{i=1}^{N} (n_i + k_i - 1) + N^2 + N$$

where $|\hat{\Sigma}_e|$ is the determinant of the error covariance matrix, and e_i represents the difference between y_i and its estimation.

- ... but the conclusions about the application of metaheuristics do not depend on the criterion used.

- For each variable it should be decided if it is included (1) or not (0) in the system: number of possible models $2^{N(N+K)}$
- Different criteria can be used to determine the goodness of the model. We use the **Akaike Information Criterion**:

$$d \cdot \ln |\hat{\Sigma}_e| + 2 \sum_{i=1}^{N} (n_i + k_i - 1) + N^2 + N$$

where $|\hat{\Sigma}_e|$ is the determinant of the error covariance matrix, and e_i represents the difference between y_i and its estimation.

- ... but the conclusions about the application of metaheuristics do not depend on the criterion used.

- For each variable it should be decided if it is included (1) or not (0) in the system: number of possible models $2^{N(N+K)}$
- Different criteria can be used to determine the goodness of the model. We use the **Akaike Information Criterion**:

$$d \cdot \ln |\hat{\Sigma}_e| + 2 \sum_{i=1}^{N} (n_i + k_i - 1) + N^2 + N$$

  where $|\hat{\Sigma}_e|$ is the determinant of the error covariance matrix, and e_i represents the difference between y_i and its estimation.
- ... but the conclusions about the application of metaheuristics do not depend on the criterion used.

Objectives

- To obtain a tool to efficiently apply and tune metaheuristics for the SEM problem

  Done: applied to model the preeclampsia, economic data...
- So, we develop unified parametrized schemes of metaheuristics

  Done: GRASP, Genetic Algorithms, Scatter Search and combinations/hybridations
- and unified parallel-parametrized schemes

  Done: in OpenMP for shared memory, with parametrized parallel functions
- Future:
  - More metaheuristics
  - Parallel schemes for other systems: message-passing and GPU
  - Hyperheuristics: autonomous selection of adequate values for the metaheuristic parameters
  - Auto-optimization: autonomous selection of adequate values for the parallelism parameters

Objectives

- To obtain a tool to efficiently apply and tune metaheuristics for the SEM problem

  Done: applied to model the preeclampsia, economic data...
- So, we develop unified parametrized schemes of metaheuristics

  Done: GRASP, Genetic Algorithms, Scatter Search and combinations/hybridations
- and unified parallel-parametrized schemes

  Done: in OpenMP for shared memory, with parametrized parallel functions
- Future:
  - More metaheuristics
  - Parallel schemes for other systems: message-passing and GPU
  - Hyperheuristics: autonomous selection of adequate values for the metaheuristic parameters
  - Auto-optimization: autonomous selection of adequate values for the parallelism parameters

Objectives

- To obtain a tool to efficiently apply and tune metaheuristics for the SEM problem

  Done: applied to model the preeclampsia, economic data...
- So, we develop unified parametrized schemes of metaheuristics

  Done: GRASP, Genetic Algorithms, Scatter Search and combinations/hybridations
- and unified parallel-parametrized schemes

  Done: in OpenMP for shared memory, with parametrized parallel functions
- Future:
  - More metaheuristics
  - Parallel schemes for other systems: message-passing and GPU
  - Hyperheuristics: autonomous selection of adequate values for the metaheuristic parameters
  - Auto-optimization: autonomous selection of adequate values for the parallelism parameters

Objectives

- To obtain a tool to efficiently apply and tune metaheuristics for the SEM problem

  Done: applied to model the preeclampsia, economic data...
- So, we develop unified parametrized schemes of metaheuristics

  Done: GRASP, Genetic Algorithms, Scatter Search and combinations/hybridations
- and unified parallel-parametrized schemes

  Done: in OpenMP for shared memory, with parametrized parallel functions
- Future:
  - More metaheuristics
  - Parallel schemes for other systems: message-passing and GPU
  - Hyperheuristics: autonomous selection of adequate values for the metaheuristic parameters
  - Auto-optimization: autonomous selection of adequate values for the parallelism parameters

Unified metaheuristic scheme

Initialize(S)
while (**not** EndCondition(S))
    SS = Select(S)
    $SS1$ = Combine(SS)
    $SS2$ = Improve($SS1$)
    S = Include($SS2$)

Facilitates working with different metaheuristics by reusing functions

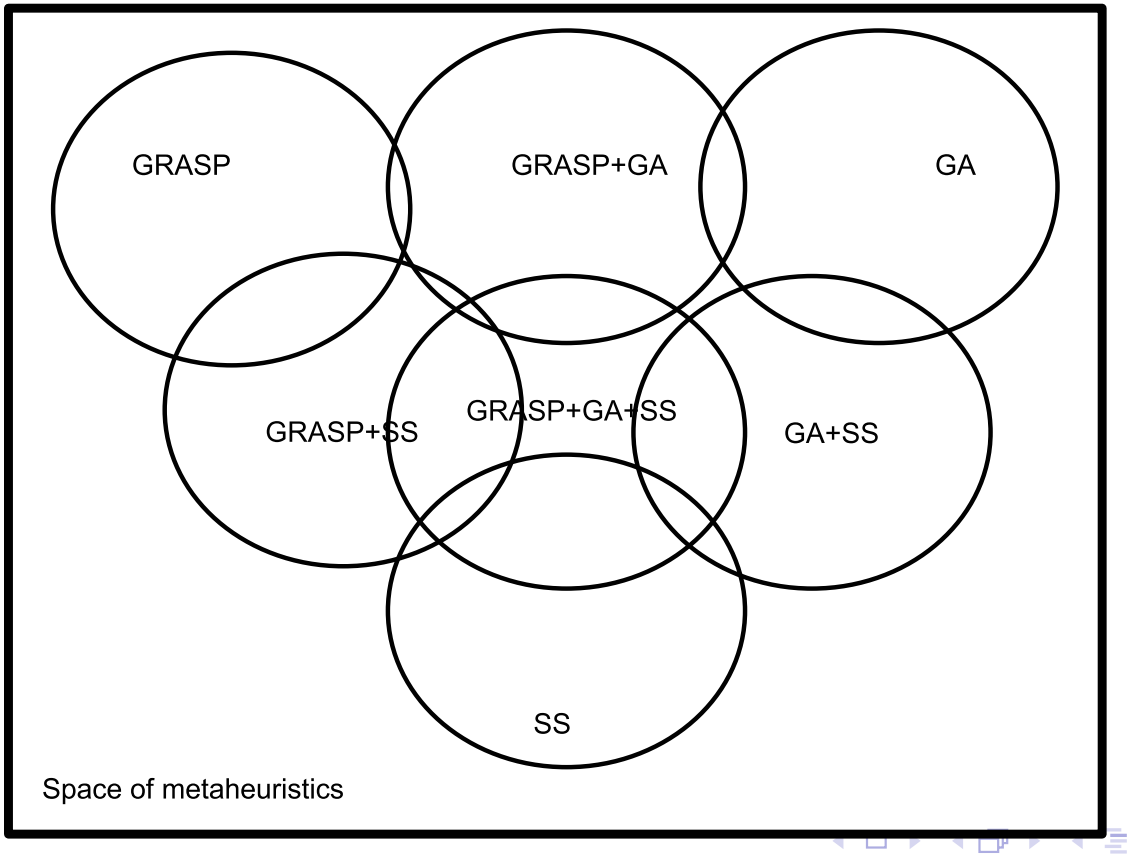
Parametrized metaheuristic scheme

Initialize(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 work with different metaheuristics/hybridation/combination by selecting different values of the parameters in the functions

Metaheuristics

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



Initialize

- 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

$$ParamIni = (INEIni, PEIIni, IIEIni, NERIni)$$

Initialize

- 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

$$ParamIni = (INEIni, PEIIni, IIEIni, NERIni)$$

Initialize

- 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

$$ParamIni = (INEIni, PEIIni, IIEIni, NERIni)$$

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 ($NWWCom$).

$$ParamCom = (NBBCom, NBWCom, NWWCom)$$

Improve

- A percentage of elements (*PEIImp*) are improved by local search..., with a certain intensification (*IIEImp*).
- A percentage of elements which are "distant" (*PEDImp*) from the reference set are generated, and an improvement is applied to these elements with a certain intensification (*IIDImp*).

$$ParamImp = (PEIImp, IIEImp, PEDImp, IIDImp)$$

Improve

- A percentage of elements (*PEIImp*) are improved by local search..., with a certain intensification (*IIEImp*).
- A percentage of elements which are "distant" (*PEDImp*) from 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 *NBEInc* elements are included in the reference set, and the others (*NERIni* − *NBEInc*) are the most "distant" from the best ones according to some distance function

$$ParamInc = (NBEInc)$$

- The method converges when a maximum number of iterations (*MNIEnd*) or a maximum number of iterations without improving the best solution (*NIREnd*) is performed

$$ParamEnd = (MNIEnd, NIREnd)$$

(they could be considered inter-metaheuristic parameters)

Include and EndCondition

- The best *NBEInc* elements are included in the reference set, and the others (*NERIni* − *NBEInc*) are the most "distant" from the best ones according to some distance function

$$ParamInc = (NBEInc)$$

- The method converges when a maximum number of iterations (*MNIEnd*) or a maximum number of iterations without improving the best solution (*NIREnd*) is performed

$$ParamEnd = (MNIEnd, NIREnd)$$

(they could be considered inter-metaheuristic parameters)

Parallel-parametrized scheme

Initialize(S,ParamIni,ThreadsIni)
while (**not** EndCondition(S,ParamEnd,ThreadsEnd))
  SS = Select(S,ParamSel,ThreadsSel)
  $SS1$ = Combine(SS,ParamCom,ThreadsCom)
  $SS2$ = Improve($SS1$,ParamImp,ThreadsImp)
  S = Include($SS2$,ParamInc,ThreadsInc)

Independent parallelization of the functions,
with parallelism parameters (number of threads) for each function.
The optimum value of the parallelism parameters depends on the values of the metaheuristic parameters (the metaheuristic or combination of 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...

Nested parallelism

```
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):
    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

```
Initialize(S,INEIni,PEIIni,IIEIni,NERIni,
                    ntIni,ntIni1,ntIni2)
while (not EndCondition(S,NMIEnd,NIREnd))
    SS = Select(S,NBESel,NWESel)
    SS1 = Combine(SS,NBBCom,NBWCom,NWWCom,
                    ntCom,ntEva)
    SS2 = Improve(SS1,PEIImp,IIEImp,PEDImp,IIDImp,
                    ntImp1,ntImp2,ntMut1,ntMut2)
    S = Include(SS2,NBEInc,
                    ntInc)
```

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.
  - `Arabi`: cluster of 102 nodes, each one with 8 cores of the processor Intel Xeon Quad-Core L5450 (one node used).

Parameters

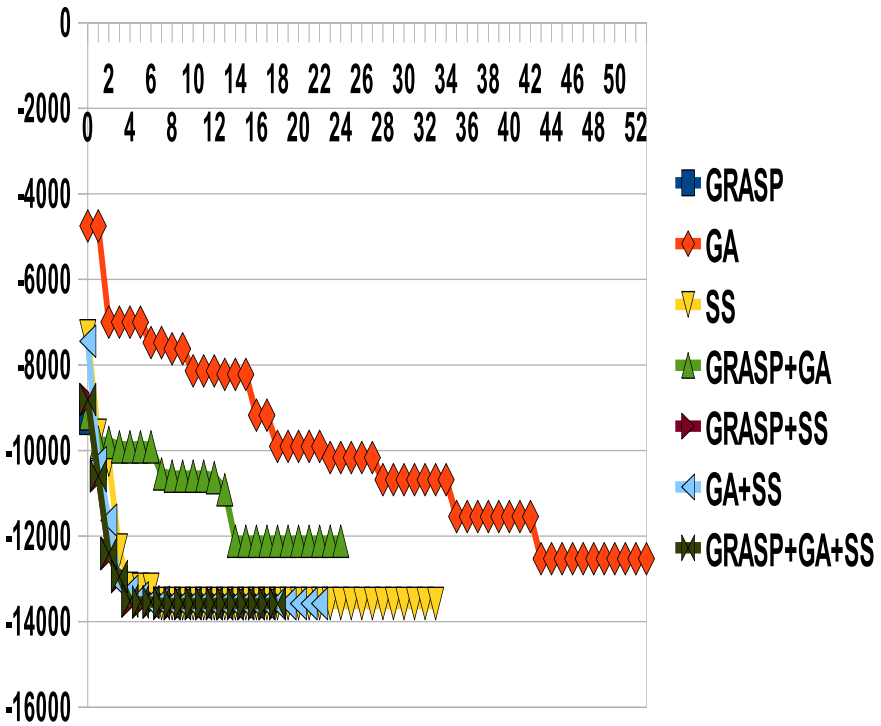
Satisfactory values for the "pure" metaheuristics obtained by experimentation

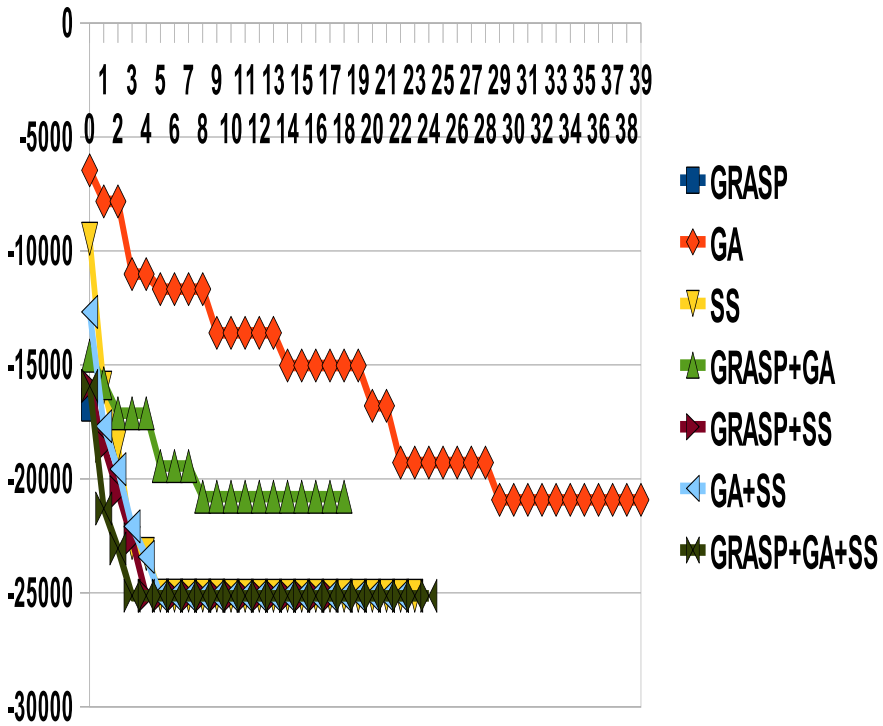
	GRASP	GA	SS	GRASP+GA	GRASP+SS	GA+SS	GRASP+GA+SS
INEIni	200	500	100	200	200	100	200
NERIni	-	500	20	200	20	50	50
PEIIni	100	0	100	100	100	100	100
IIEIni	10	-	10	10	10	10	10
NBESel	-	500	10	200	10	25	25
NWESel	-	0	10	0	10	25	25
NBBCom	-	250	90	100	90	90	90
NBWCom	-	-	100	-	100	100	100
NWWCom	-	-	90	-	90	90	90
PEIImp	-	0	100	0	100	100	100
IIEImp	-	-	5	-	5	5	5
PEDImp	-	10	0	10	0	10	10
IIDImp	-	0	-	0	-	5	5
NBEInc	-	500	10	200	10	25	25

The results previously obtained with Genetic algorithms are improved

Value of AIC in successive iterations, $N = K = 20$

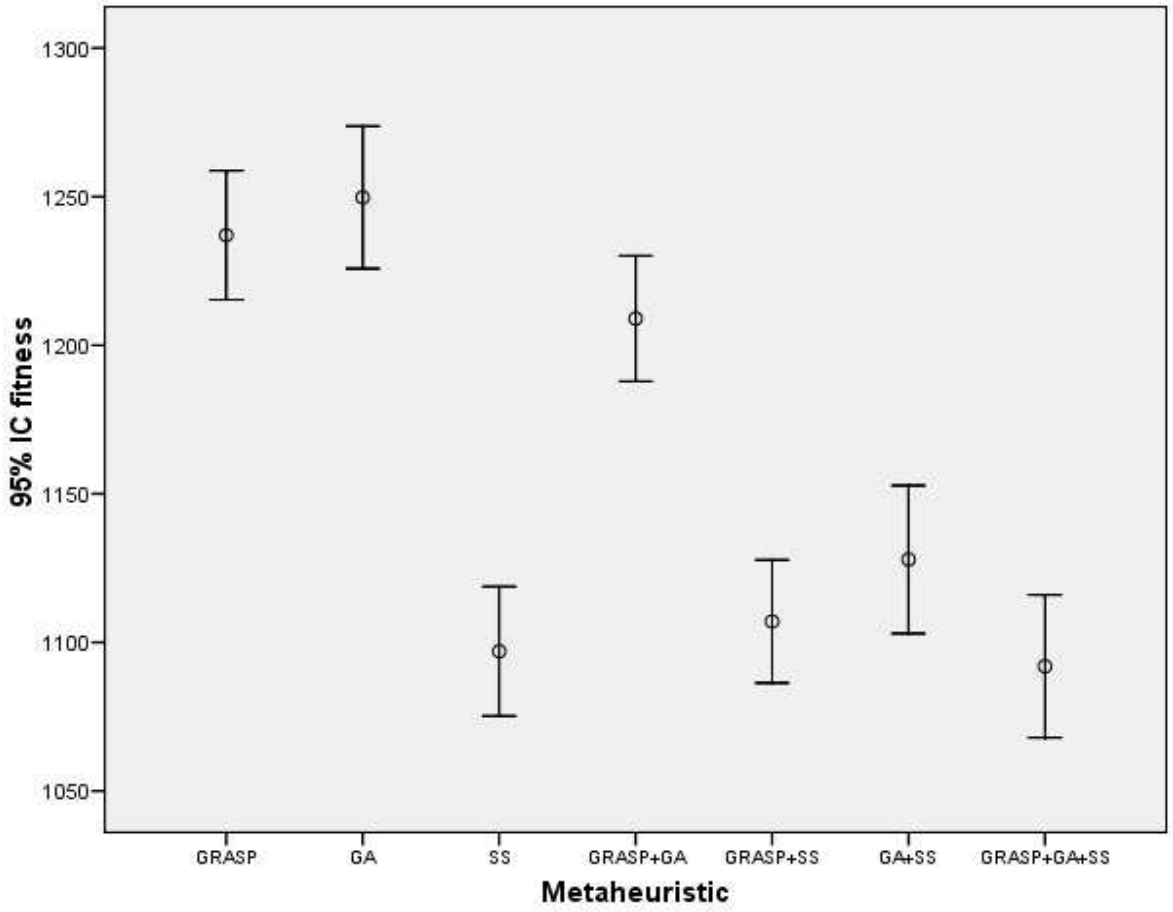
$d = 100$, $d = 200$



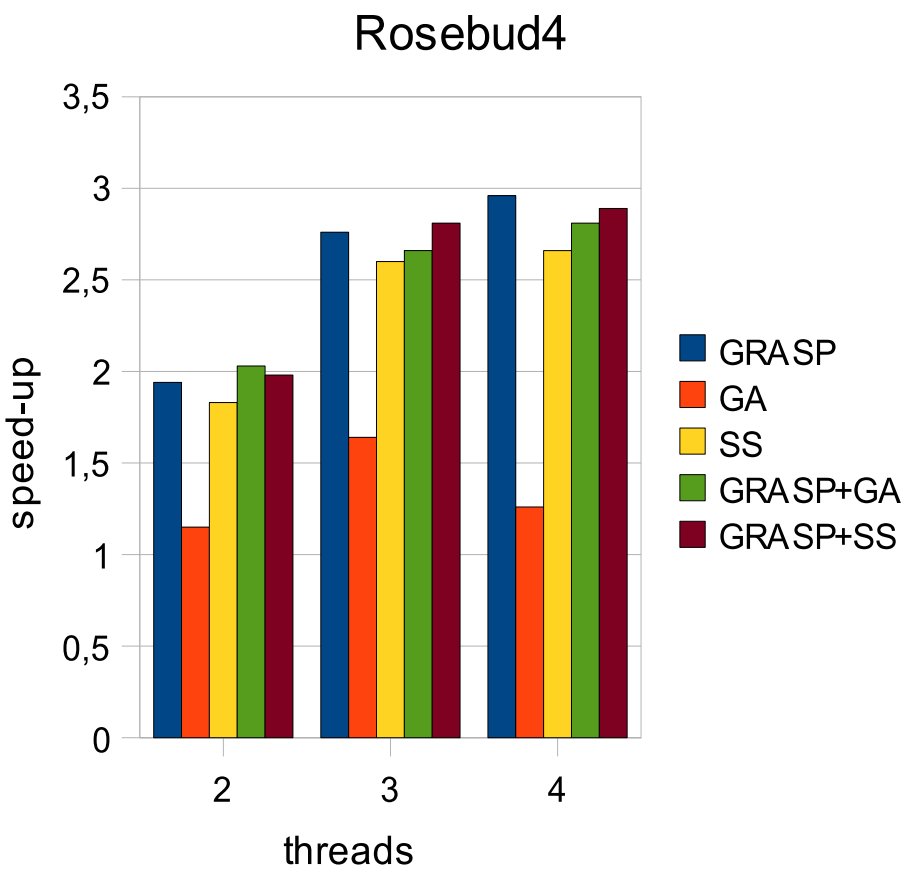


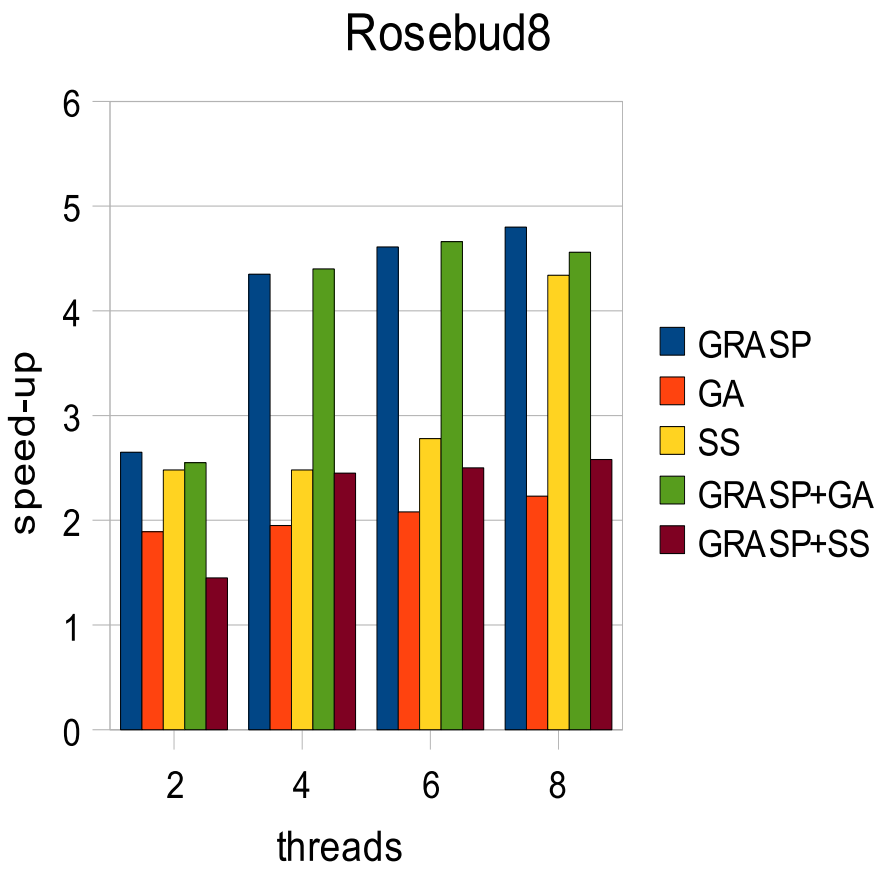
Statistical study of the influence of the parameters can be carried out

700 combined metaheuristics are generated, 100 in the environment of each basic metaheuristic in the space of metaheuristics.



Speed-up obtained for different metaheuristics in Rosebud.
$N = K = 20,\ d = 100$

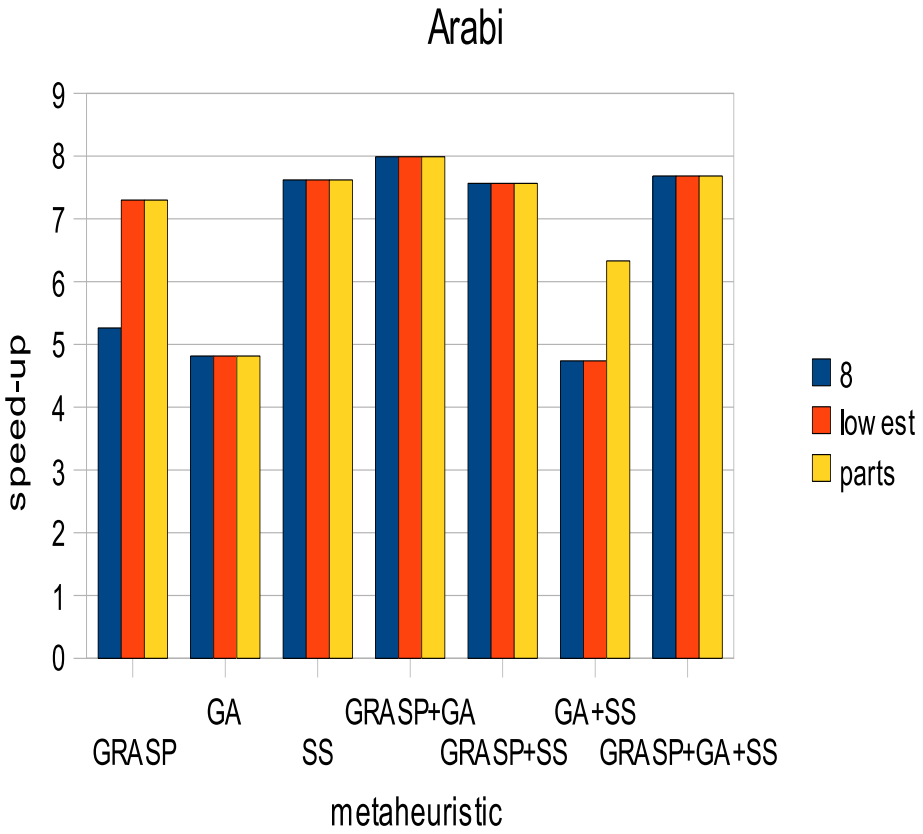


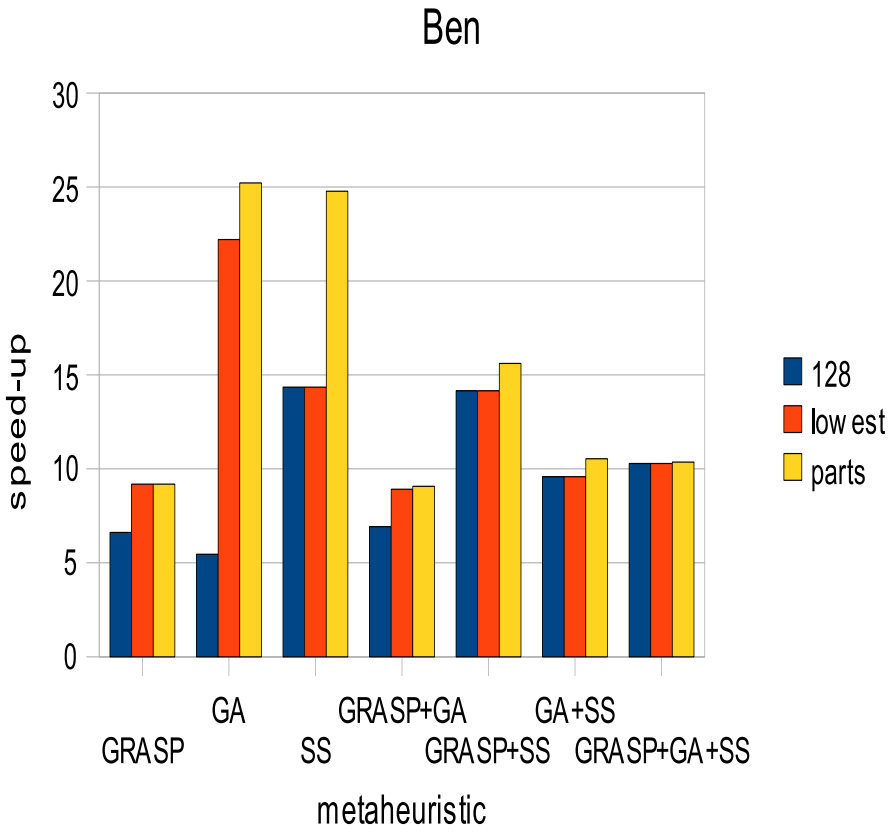


Use of parallelism parameters

speed-up with the maximum number of cores, the optimum number of threads for the complete program, and different numbers of threads in the different functions and levels

$N = K = 20,\ d = 100$





Conclusions

- The use of parameterized metaheuristics allows us:
  - To experiment with different metaheuristics and combinations (inter- parameters) to obtain a satisfactory one 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
- The methodology is being applied to other problems: tasks-to-processors assignation, p-hub, signal filters design, electricity consumption in water wells

Future research

- Inclusion of more "pure" metaheuristics
- Design of hyperheuristics to automatically select the values of the inter- parameters 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...)