A parameterised shared-memory scheme for parameterised metaheuristics

Francisco Almeida, Domingo Giménez, Jose J. López-Espín

falmeida@ull.es, domingo@um.es, jlopez@umh.es

Universidad de La Laguna

Miguel Hernández

Motivation

Parallel systems are formed by multicore ⇒ develop efficient multicore versions of our algorithms, with OpenMP.

A unified parameterised scheme for metaheuristics, facilitates the easy development of new metaheuristics or hybrid metaheuristics, for experiment and adaptation to a particular problem:

```
Initialize(S)
while (not EndCondition(S))
  SS = Select(S)
  if(|SS| > 1) SS1 = Combine(SS)
  else SS1 = SS
  SS2 = Improve(SS1)
  S = Include(SS2)
```

To obtain a well-tailored metaheuristic for a problem, experiment with a large number of metaheuristics and their parameters: the time dedicated to the experiments is very large ⇒ parallel programming.

Common development of parallel versions: from the unified metaheuristic scheme to obtain a unified parallel scheme for metaheuristics.

The parallel scheme is parameterised: the values of some algorithmic parameters can be selected to optimise the execution of the parallel metaheuristic. The optimum values of the algorithmic parameters depend of those of the metaheuristic parameters and of the characteristics of the computational system.

Parameterised metaheuristics

Each basic function in the unified metaheuristic scheme can be parameterised: different values of the parameters give different metaheuristics, hybridation/combination of metaheuristics or different versions of a particular metaheuristic.

```
Initialize(S,ParamInit)
while (not EndCondition(S,ParamEndCond))
  SS = Select(S,ParamSelec)
  if(|SS| > 1)
    SS1 = Combine(SS,ParamComb)
  else SS1 = SS
  SS2 = Improve(SS1,ParamImpr)
  S = Include(SS2,ParamIncl)
```

An example to combine GRASP, Genetic Algorithms and Scatter Search:

- `ParamInit` = (size of initial reference set;number of elements to improve;intensity of the improvement;number of elements in the reference set)
- `ParamEndCond` = (maximum number of iterations;maximum number of iterations without improvement)
- `ParamSelect` = (number of best elements to select;number of worst elements to select)
- `ParamCombi` = (number of best-best combinations;number of best-worst combinations;number of worst-worst combinations)
- `ParamImpr` = (percentage of generated elements to improve;intensity of the improvement of generated elements;percentage of elements to modify;intensity of the improvement of elements obtained by modification)
- `ParamIncl` = (number of best elements to include in the reference set;number of worst elements to include in the reference set)

Parameterised SM scheme

The functions are parallelised independently. Different parallel patterns should be identified in the basic functions.

Two basic parallel schemes:

- The elements in a set are treated independently:

```
1loop(Param):
    omp_set_num_threads
     (loop-thr(Param))
   #pragma omp parallel for
     loop in elements
       treat element
```

  The set of metaheuristic parameters (`Param`) is passed to the function, and the number of threads to be used in the parallel loop (`loop-thr`) is obtained as a function of the values of the metaheuristic parameters.

- Two-level parallelism scheme:

```
2level(Param):
    omp_set_num_threads
     (level1-thr(Param))
   #pragma omp parallel for
     loop in elements
       level2(Param,level1-thr)

level2(Param,level1-thr):
    omp_set_num_threads
     (level2-thr(Param,level1-thr))
   #pragma omp parallel for
     loop in elements
       treat element
```

  The number of threads in the first level (`level1-thr`) is obtained as a function of the parameters of the metaheuristic (also of its functions, and consequently of the cost of them in the computational system).

  The number of threads to work in the second level (`level2-thr`) is obtained as a function of the metaheuristic parameters and the number of threads working in the first level.

Experiments - Problem

Problem used in the experiments: given a set of values (obtained by experimentation, survey...) to obtain the best Simultaneous Equation Model (SEM) which best represents the variables dependences.

SEM are developed by experts with a wealth of experience in the particular problem represented by the model.

An automatic tools to provide the experts with satisfactory models is interesting when the dependence of the variables is not clear or when experiments are being carried out to determine variables to be included in the model.

Necessary to evaluate a large amount of candidate models ⇒ use of metaheuristics.

The quality of the model can be measured with different criteria (Akaike Information Criterion, AIC).

An element is defined as a matrix. In each row, an equation is represented using ones and zeros. If variable j appears in equation i, the value for the (i, j) position is one, and zero if not.

Experiments - Systems

Experiments carried out in the supercomputer `BenArabi` of the Supercomputing Centre of the Fundación Parque Científico of Murcia.

`Ben` is a HP Integrity Superdome SX2000 with 128 cores of the processor Intel Itanium-2 dual-core Montvale.

`Arabi` is a cluster of 102 nodes, each one with 8 cores of the processor Intel Xeon Quad-Core L5450.

Experiments in individual nodes: 8 and 128 cores.

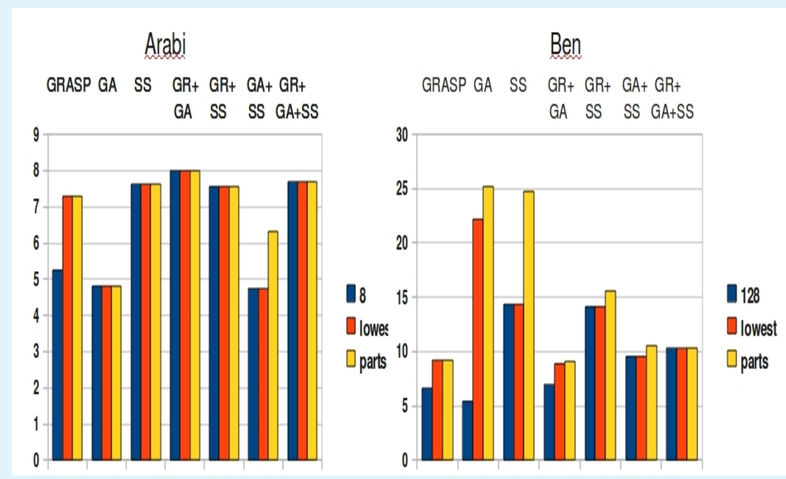
Experiments - Metaheuristics

	GR	GA	SS	GR+GA	GR+SS	GA+SS	G+G+S
#Ele.Ini.	200	500	100	200	200	100	200
#Ele.Ite.	-	500	20	200	20	50	50
#Ele.Mej	100	0	100	100	100	100	100
Int.Mej.	10	-	10	10	10	10	10
#Best	-	500	10	200	10	25	25
#Worst	-	0	10	0	10	25	25
#B-B	-	250	90	100	90	90	90
#B-W	-	-	100	-	100	100	100
#W-W	-	-	90	-	90	90	90
#Ele.Mej.	-	0	100	0	100	100	100
Int.Mej.	-	-	5	-	5	5	5
#Ele.Mod.	-	10	0	10	0	10	10
Int.Mod.	-	0	-	0	-	5	5
#Best.Inc.	-	500	10	200	10	25	25

Experiments - Results

The figure shows the speed-up achieved in the two systems with the seven metaheuristics:

- 8 or 128: using the maximum number of cores in the systems (8 in `Arabi` and 128 in `Ben`) without nested parallelism.
- lowest: with the number of cores in each parallelism level which gives the lowest execution time.
- parts: with the best combination of threads in the initialisation part and in the iteration part.



`Arabi`: speed-up is close to the number of cores, and normally the best configuration is to use non nested parallelism and 8 cores. In the metaheuristics with lowest execution time per iteration the speed-up is lower because the sequential time of the parallelised parts is very low.

`Ben`: to use the maximum number of cores is not a good option, and some strategy to select the number of threads to work on the solution of the problem is preferable. The speed-up is always far from the maximum achievable.

Furthermore, the randomness in the execution in the metaheuristics makes it difficult to draw definitive conclusions, but experiments with other problem sizes and configurations confirm this behaviour.

Funding

This work has been partially supported by the Conserjería de Educación de la Región de Murcia (Fundación Séneca, 08763/PI/08), and by the Ministerio de Ciencia e Innovación (TIN2008-06570-C04).