pALS: an Object Oriented Framework for Developing Parallel Cooperative Metaheuristics

Harold Castro, Andrés Bernal
COMIT (COMunications and Information Technology)
Systems and Computing Engineering Department
Universidad de los Andes
Bogotá, Colombia

Agenda

- Motivation and Related Works
- ALS
- The pALS framework
- Achievements and Results
- Conclusions
Motivation

- Research on Optimization (COPA Research Group)
- JG2A: Java Framework for Genetic Algorithms on a Grid
  - Evolution for JGA
    - Instances parallelization
    - Individual parallel evaluation
  - Uses standard tools: Condor, Globus, Java
  - Allows different models: Master/Slave, Cellular, Island, Hierarchical
- Good results, but
  - Only works for GA
  - Lacks of statistics

NIDISC’10, Atlanta, April 19-23
### Current frameworks

<table>
<thead>
<tr>
<th></th>
<th>Sequential</th>
<th>Parallel</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Single Metaheuristic</strong></td>
<td>JGA, SFERES, ECJ 18</td>
<td>JG2A, GALib</td>
</tr>
<tr>
<td><strong>Metaheuristics</strong></td>
<td>HotFrame, EasyLocal, iOpt, PISA, Templar</td>
<td>ParadisEO, HeuristicLab</td>
</tr>
</tbody>
</table>

### Candidates

**ParadisEO**
- C++
- Templates
- Four different modules
  - Paradis-EO
  - Paradis-MO
  - Paradis-MOEO
  - Paradis-PEO
    - GT4 + MPI
- Difficult to use

**HeuristicLab**
- C#
- Multiple algorithms
  - GA
  - Evolutionary strategies
  - Ant Colony
  - Tabu search
- No grid enabled
- Security
- Heterogeneity
The ALS framework

- ALS, Adaptive Learning Search provides a conceptual frame to analyze Metaheuristics
- Based on the operator concept
  - Any specific operation
  - Represents an algorithm, a set of algorithms or a single Metaheuristic operation
- 4 operators are common to any Metaheuristic

Initialize
Iterate <while non stop condition>
  Sampling, selects elements
  Learning, extract information
  Diversification, search for new solutions
  Intensification, try to improve current solution
  Replace, replace the old solution
End

Validating ALS

<table>
<thead>
<tr>
<th>Simulated Annealing</th>
<th>Variable Neighborhood Search</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALS</td>
<td>ALS</td>
</tr>
<tr>
<td>Sampling</td>
<td>Implicit</td>
</tr>
<tr>
<td>Learning</td>
<td>Learning</td>
</tr>
<tr>
<td>Diversification</td>
<td>Diversification</td>
</tr>
<tr>
<td>Intensification</td>
<td>Intensification</td>
</tr>
<tr>
<td>Root</td>
<td>Neighbourhood Change</td>
</tr>
<tr>
<td>Tabu Search</td>
<td>Local Search</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ALS</th>
<th>Tabu Search</th>
<th>Genetic Algorithms</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALS</td>
<td>Sampling</td>
<td>Implicit</td>
</tr>
<tr>
<td>ALS</td>
<td>Learning</td>
<td>Learning</td>
</tr>
<tr>
<td>ALS</td>
<td>Diversification</td>
<td>Diversification</td>
</tr>
<tr>
<td>ALS</td>
<td>Intensification</td>
<td>Intensification</td>
</tr>
<tr>
<td>ALS</td>
<td>Local Search</td>
<td>Selective</td>
</tr>
</tbody>
</table>

NIDISC’10, Atlanta, April 19-23
pALS proposal

To develop a framework making easy the implementation of different metaheuristics on parallel environments, helping with:

- Parallel design
- Callback services
- Statistic services
- Cooperative strategies

NIDISC’10, Atlanta, April 19-23

pALS architecture

NIDISC’10, Atlanta, April 19-23
- Contains classes for modeling metaheuristics
- Solution transformation based process
- Most classes are abstract classes
- Behavior defined by configuration files
  - Define operators and their parameters
  - Every operator has its own configuration file (or inherit from its parent)
  - Makes pALS a white box framework

**pALS Core Layer**

- Represents a transformation of a set of solutions
- May be used to model complex (i.e. metaheuristics) or simple (i.e. crossover operator) operations
- An execution is a hierarchy of operators

---

**Operator**

- pALS
- Generic Algorithms
  - Mutation
  - Crossover
  - Evaluation
  - Selection

---

**Solution**

- Representation
- Results

---

**Solutions**

- Operator
  - ListOperator
  - ObjectOperator

- Select best n solutions
- Flip bits in a solution
Cooperator $i$ takes solutions from $(i-1)\%m$

- Common with small $m$

**Ring Topology Cooperator**

**Callback**
- Defines conditional behaviors after ending an operator execution
- Defined as a condition and an action

**Statistic**
- About transformations inside operators (time, best solution, etc.)
- Bounded by the InitStatistic and EndStatistic methods

**Context Builder**
- Allows to use external parameters from a file or an external Java class

**Support classes**

- value = 10
- vector = 1 2 3 4
- matrix = 1 2 3 4, 5 6 7 8
• Captures operators’ execution logic

• Offers flexibility and transparency to the execution (centralized, distributed, etc.)

• Every operator has its own ExecutionPrototype defined by the user in a configuration file

• The ExecutionPrototype class can be easily extended

ExecutionLayer

Thread/Grid delegates

• Takes advantage of multi-core platforms

• Launch a new thread for the operator’s execution

• Manages security

• Transparent for users
Execution models

- Multi-Boot
- Parallel Execution
- Mix

Life cycle

- Global Settings
- pALs
  - Operator
    - Settings
    - Cooperate
    - Context Builder
    - Init Statistics
    - Execution Prototype
    - End Statistics
    - Callback
    - Cooperate
  - Publish

NIDISC’10, Atlanta, April 19-23
pALS offer

Flexibility
- Settings

Functionality
- Operator
- Cooperator
- Context Builder
- Callback
- Statistic

pALS

Performance
- Execution
  - Prototype
- Thread
  - Delegate
- Local
  - Delegate
- Grid
  - Delegate

Wide scope / reusability
- Genetic Algorithm
- Tabu Search
- Simulated Annealing
- VNS

pALS experimentation

Boolean satisfiability problem (SAT)
Tabu Search

Knapsack
Genetic Algorithm with island deployment
Conclusions

- pALS is a framework implementing the metaheuristic concepts behind ALS
- Not a set of metaheuristics
- Allows for different parallel execution, transparent to users
- White-box framework, ready to be used
- Enhanced tools for research: callbacks, statistics
- Self-incrementing
- Public available at http://sistemas.uniandes.edu.co/~comit
- Needs further testing and comparisons

Configuration file example

```
# SAT Tabu Search Algorithm settings file
execution_instance = BasicTabuSearch
objectives = max
representation_length = 50
execution_instance_settings_file = data/satisfiability.properties
neighborhood_size = 200
 taboo_list_length = 5
 representation = BinaryArrayRepresentation
 init_solution = RandomPopulationGenerator
 neighborhood = SinglePointFlipOperator
 binary_flip_operator_rate = 0.5
 record_tabu = ArrayIndexesTabuRecordOperator
 taboo_check = ArrayIndexesTabuCheckOperator
 evaluation = CNFSatisfiabilityOperator
 selection = BestSolutionsSelectionOperator
 solutions = 1
 max_iterations = 1
cnf_file = data/aim-50-2_0-yes1-1.cnf
```
Grid delegate execution