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

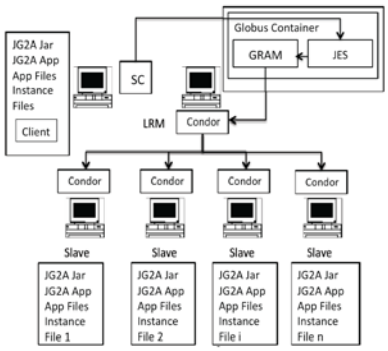
Universidad de los Andes

- 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

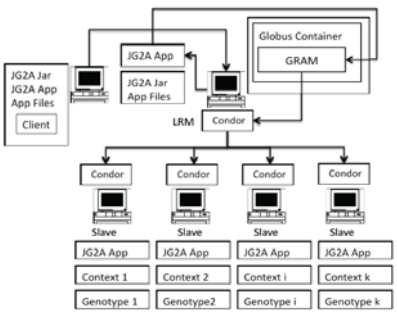
JG2A

Universidad de los Andes

Parallel instances



Parallel Evaluation



Current frameworks

Universidad de los Andes

	Sequential	Parallel
Single Metaheuristic	JGA, SFERES, ECJ 18	JG2A, GALib
Metaheuristics	HotFrame, EasyLocal, iOpt, PISA, Templar	ParadisEO, HeuristicLab

Candidates

Universidad de los Andes

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

Universidad de los Andes

- 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

Universidad de los Andes

Simulated Annealing

ALS	
Sampling	Directo
Learning	Mecanismo estocástico entre el conjunto de puntos y la temperatura
Diversification	Monotono de la función objetivo a través del método de la Metrópolis.
Intensification	Disminución de la temperatura

Variable Neighborhood Search

ALS	
Sampling	Implícito
Learning	Sistema
Diversification	Neighborhood Change
Intensification	Local Search

Tabu Search

ALS	
Sampling	Directo
Learning	Tabu List and direct sampling of the solution
Diversification	Solution change
Intensification	Local Search

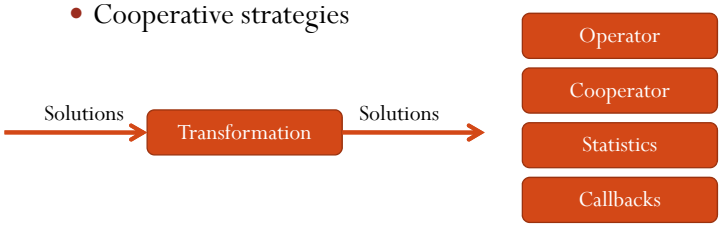
Genetic Algorithms

ALS	
Sampling	Implícito
Learning	Cruce
Diversification	Mutación
Intensification	Selección

pALS proposal

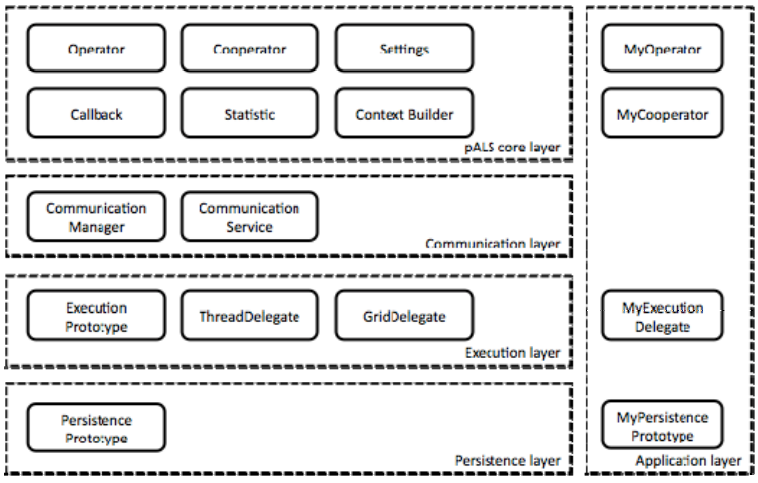
Universidad de los Andes

- To develop a framework making easy the implementation of different metaheuristics on parallel environments, helping with
  - Parallel design
  - Callback services
  - Statistic services
  - Cooperative strategies



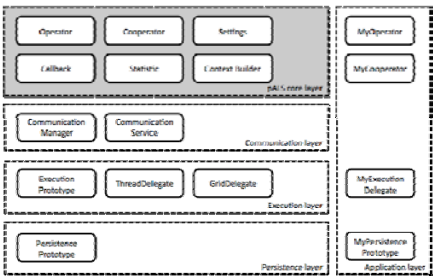
pALS architecture

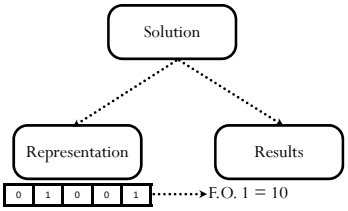
Universidad de los Andes



- 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 of pALS a white box framework

Universidad de los Andes

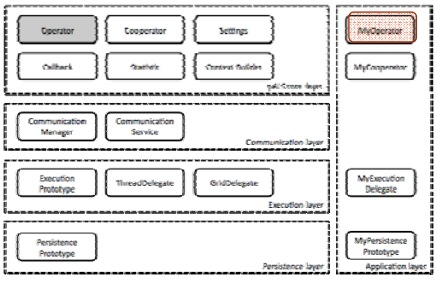


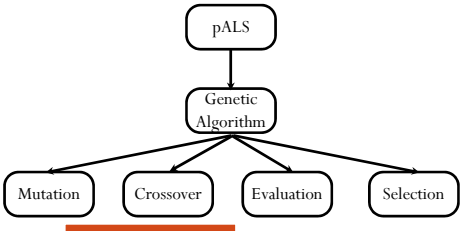


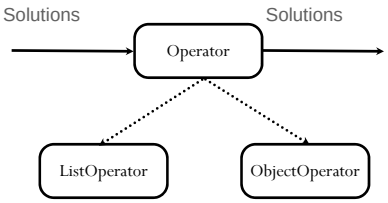
pALS Core Layer

- 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

Universidad de los Andes

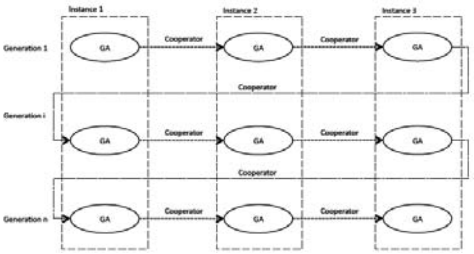


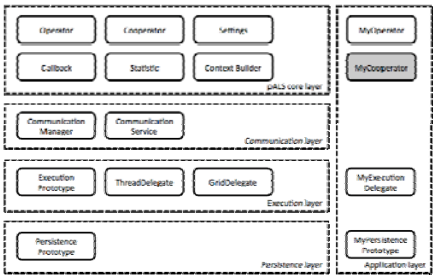


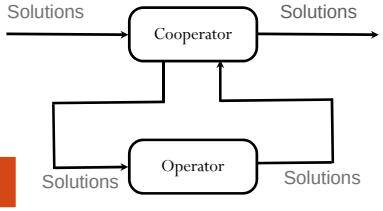


Operator

- Cooperator *i* takes solutions from *(i-1)%m*
  - Common with small *m*







Ring Topology Cooperator

Callback

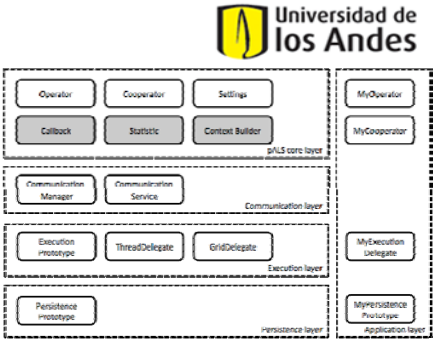
- Defines conditional behaviors after ending an operator execution
- Defined as a condition and an action

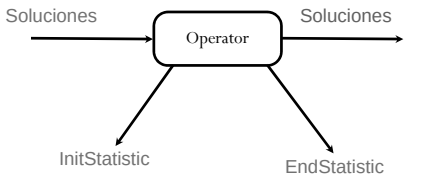
Statistic

- 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

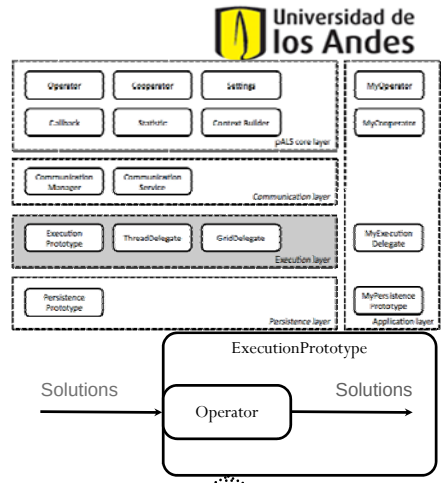


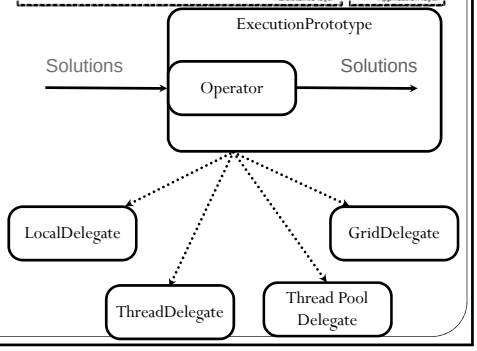



```
value   =   10
vector  =   1 2 3 4
matrix  =   1 2 3 4 ,\
            5 6 7 8
```

Support classes

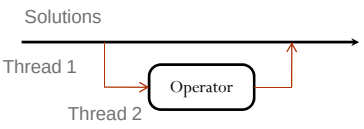
- Captures operators' execution logic
- Offers flexibility and transparency to the execution (centralized, distributed, etc.)
- Every operator has its own ExecutionPrototype defined by the user in a configuration file
- The ExecutionPrototype class can be easily exteneded



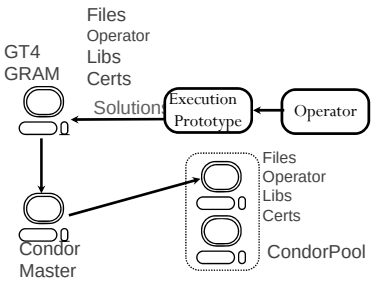


ExecutionLayer

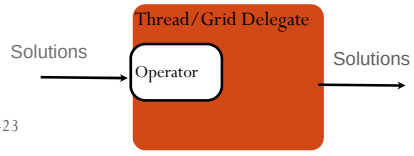
Thread/Grid delegates



- Takes advantage of multi-core platforms
- Launch a new thread for the operator's execution

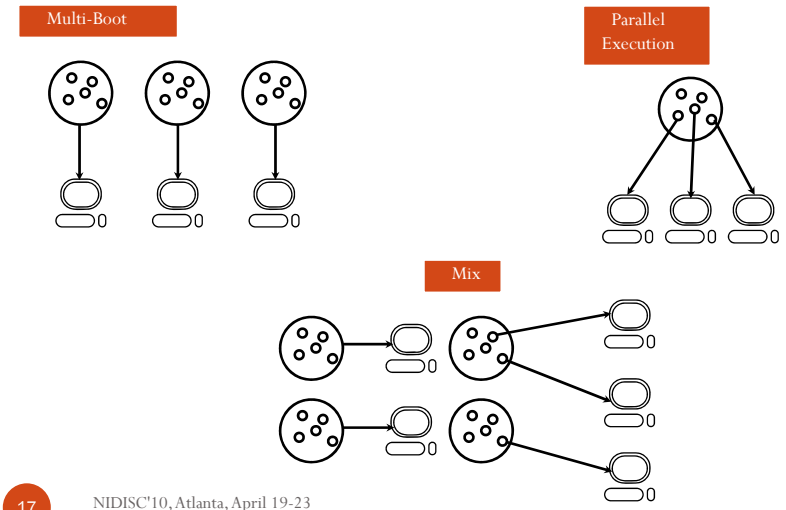


- Manages security
- Transparent for users



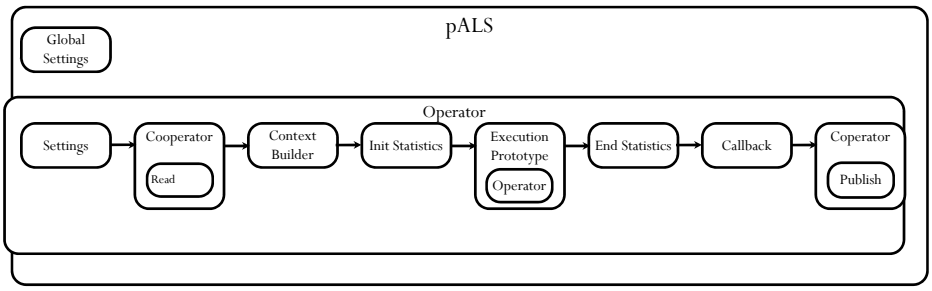
Execution models

Universidad de los Andes



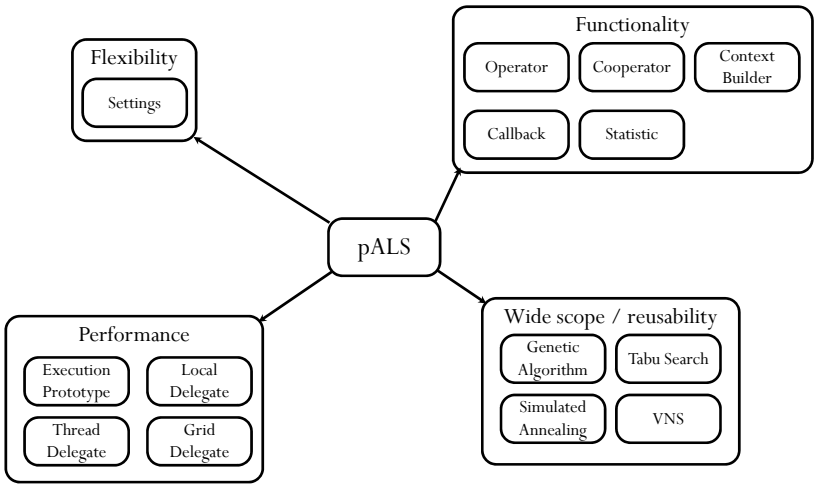
Life cycle

Universidad de los Andes



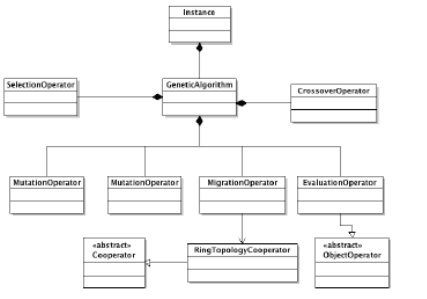
pALS offer

Universidad de los Andes



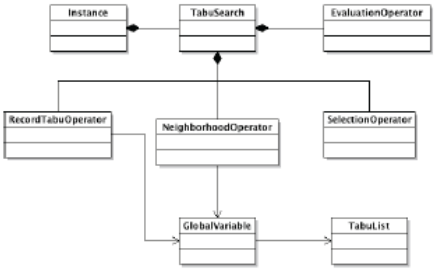
pALS experimentation

Universidad de los Andes



Knapsack
Genetic Algorithm with island deployment

Boolean satisfiability problem (SAT)
Tabu Search



Conclusions

Universidad de los Andes

- 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

Universidad de los Andes

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

Grid delegate execution

Universidad de los Andes