Memetic Algorithms for the Automatic Discovery of Software Architectures

Aurora Ramírez, Rafael Barbudo, José Raúl Romero, Sebastián Ventura

Dept. Computer Science and Numerical Analysis

University of Córdoba, Spain

16th Int. Conf. on Intelligent Systems Design and Applications (ISDA)

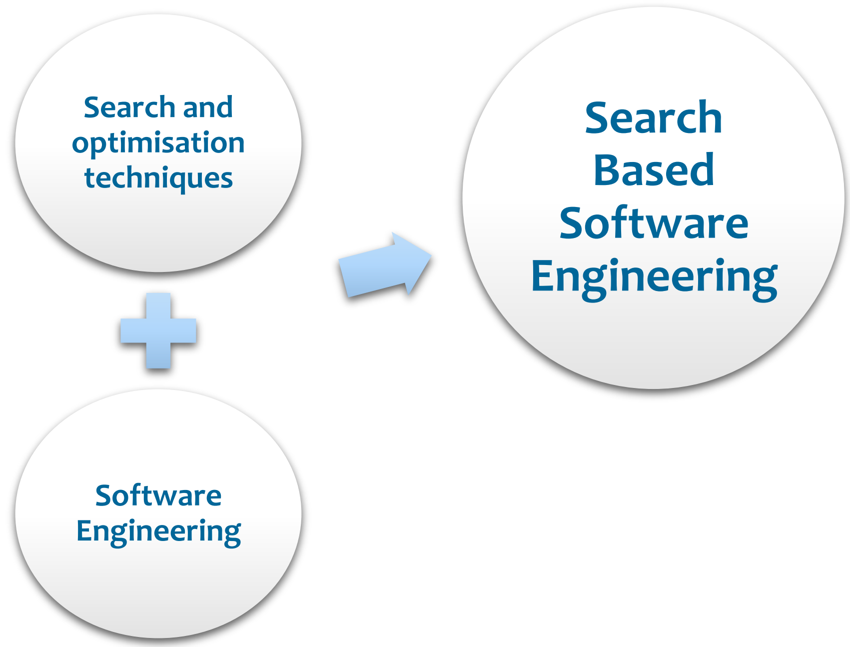
December 14-15, 2016 – Porto (Portugal)

Contents

Introduction

Search Based Software Engineering

- **Search Based Software Engineering (SBSE)**
  - Applying metaheuristics to solve Software Engineering tasks
  - Requirements prioritisation, generation of test cases...



- More specifically... SBSD
  - Automatic exploration of design alternatives
  - SBSE as a supporting mechanism for software architects

Introduction

Evolutionary discovery of architectures

We want to automatically identify the component-based architecture of a system from its analysis model (represented as a class diagram)

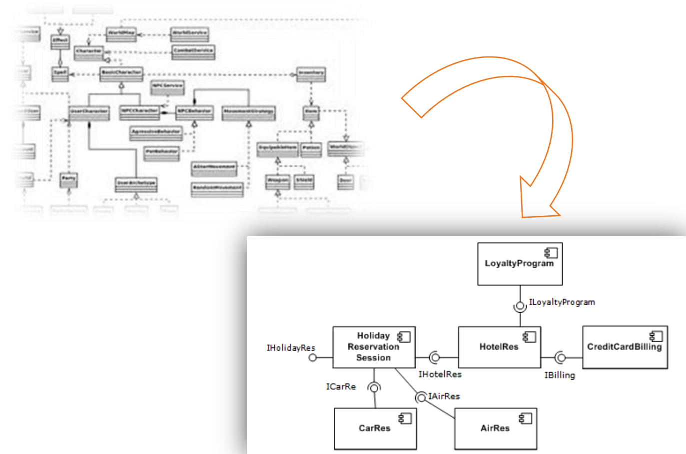
A flexible and comprehensive tree encoding

A fitness function based on rankings of design metrics

EA

A mutation operator able to perform 5 architectural transformations

An adaptive mechanism to deal with constraints



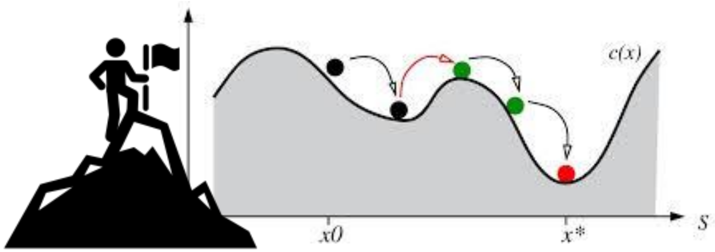
Metric	Formula
ICD: Intra-modular Coupling Density	$ICD_i = ((\#cl_t - \#cl_i)/\#cl_t) \cdot (CI_i^{in}/(CI_i^{in} + CI_i^{out}))$ $ICD = \sum_{i=1}^{n} ICD_i/n$
ERP: External Relations Penalty	$ERP = \sum_{i=1}^{n} \sum_{j=i+1}^{n} (w_{as} \cdot n_{as_{ij}} + w_{ag} \cdot n_{ag_{ij}}$ $+ w_{co} \cdot n_{co_{ij}} + w_{ge} \cdot n_{ge_{ij}})$
CS: Critical size	$CS = \sum_{i=1}^{n} CC_i,\ CC_i = 1\ if\ \#cl_i > threshold,\ 0\ otherwise$
CB: Component Balance	$SB(n) = \frac{n-\gamma}{\mu-\gamma}\ if\ n < \mu\ ,\ = 1 - \frac{n-\mu}{\omega-\mu}\ if\ \mu < n < \omega\ ,\ = 0\ if\ n \geq \omega$ $CSU(n) = 1 - Gini(\{\#cl_i \forall i \in [1,n]\})\ ,\ CB = SB(n) \cdot CSU$

RQ: In which ways can local search enhance the evolutionary discovery of software architectures?

Introduction
Memetic algorithms

Evolutionary algorithm (EA) + **Local search (LS)** = **Memetic algorithm (MA)**

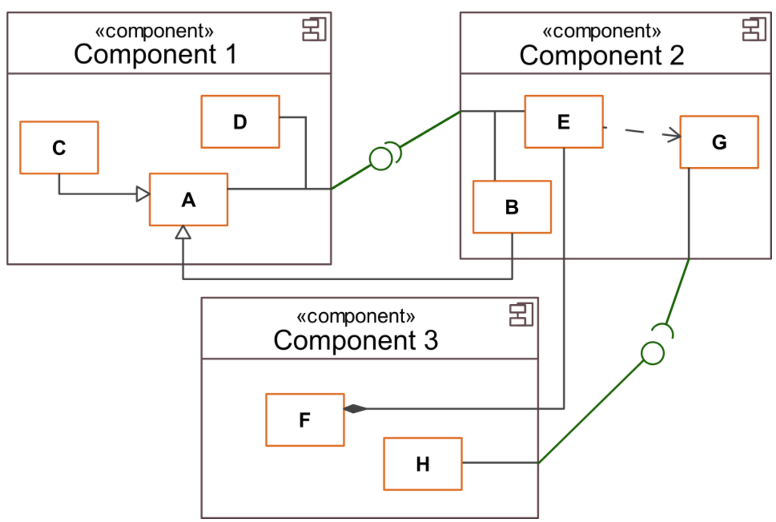
- **Local search** techniques
  a. Hill climbing (HC)
  b. Simulated annealing (SA)



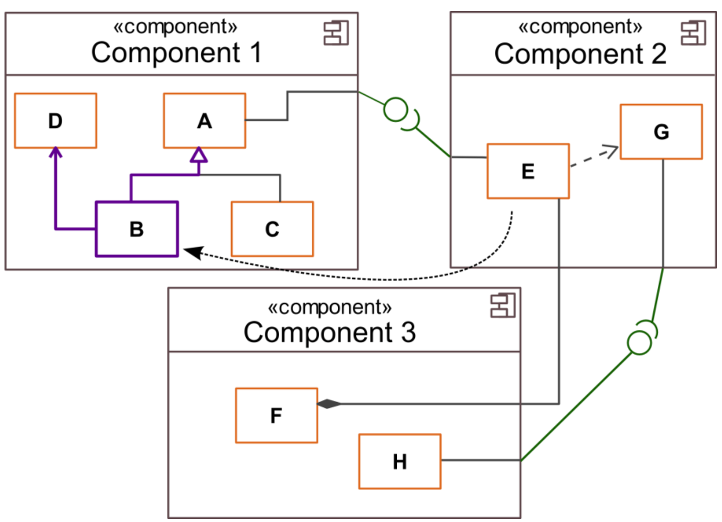
- **Design decisions**:
  1. Trade-off between exploration and exploitation
  2. Which LS algorithm is more effective?
  3. When and how often should LS be applied?
  4. Which solutions from the EA should be selected?

Proposed memetic algorithms
Local search procedure

- Goal: To explore the **neighbourhood** of a given solution
- Use of local search:
  - 1 neighbour is generated by **moving 1 class** (random)
  - The **acceptance criterion** considers the whole population



Initial solution

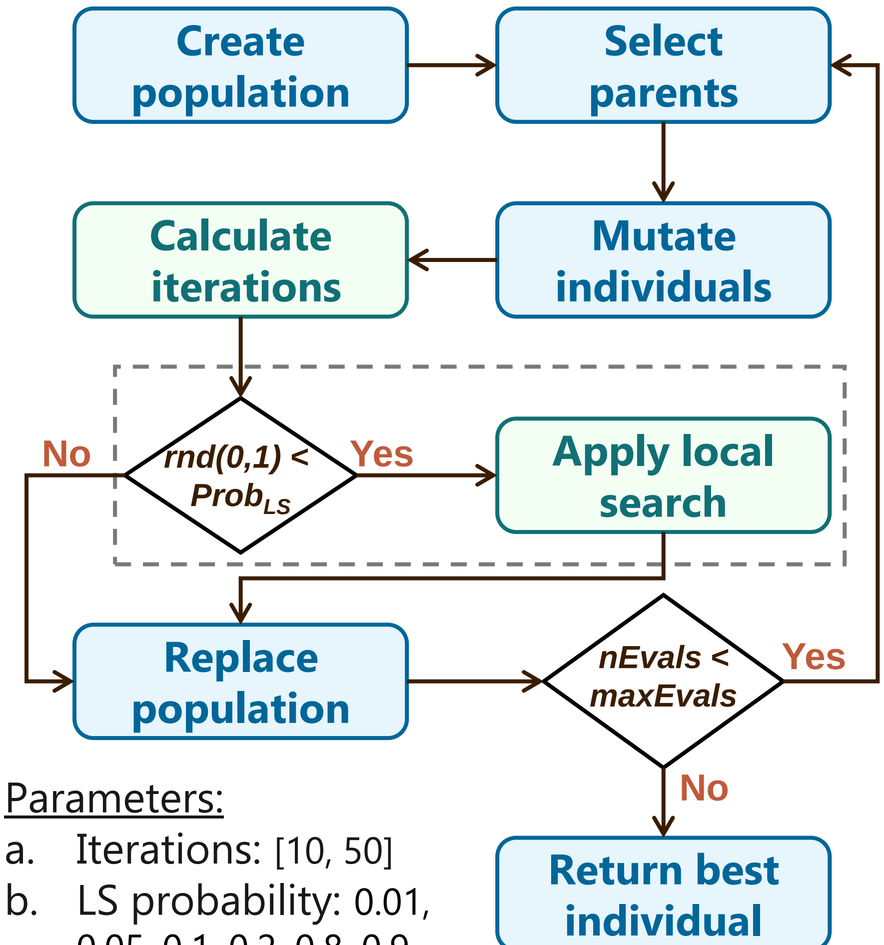


Neighbour solution

Proposed memetic algorithms

EA(LS) and EA+LS

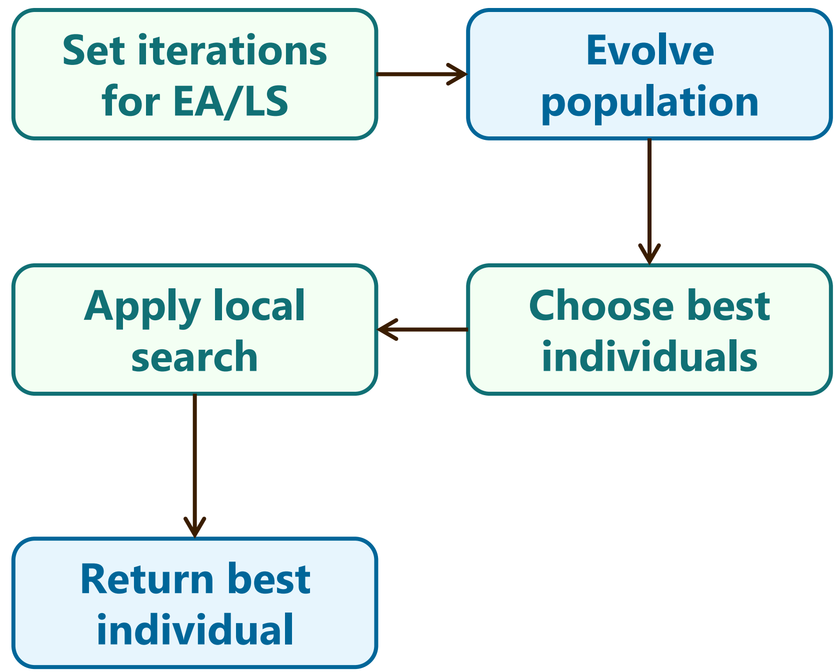
EA(LS) : Local search as genetic operator



Parameters:

a. Iterations: [10, 50]
b. LS probability: 0.01, 0.05, 0.1, 0.2, 0.8, 0.9, 0.95, 0.99, 1.0

EA+LS: Local search as post-processing



Parameters:

a. Iterations: 50, 100, 200
b. Solutions (%): 10, 15, 20

Experiments and results

Performance of local search

EA(LS)

Algorithm	% Satisfactory movements	Number of evaluations
EA	-	23615.90 ± 732.20
EA(HC)_0.001	35.60 ± 3.20	23589.80 ± 663.80
EA(HC)_0.05	21.60 ± 2.90	23246.60 ± 1137.20
EA(HC)_0.10	21.30 ± 2.50	23067.20 ± 1473.70
EA(HC)_0.20	20.80 ± 2.90	22462.40 ± 2493.20
EA(HC)_0.50	20.10 ± 2.80	22421.80 ± 2431.00
EA(HC)_0.80	19.50 ± 3.00	22284.00 ± 2813.70
EA(HC)_0.90	19.47 ± 3.00	22535.70 ± 2718.10
EA(HC)_0.95	19.50 ± 3.10	21960.60 ± 3038.00
EA(HC)_1.00	19.40 ± 3.00	22460.60 ± 2801.20
EA(SA-m)_0.001	35.20 ± 3.00	23616.00 ± 677.30
EA(SA-m)_0.05	21.90 ± 2.70	23272.10 ± 1112.40
EA(SA-m)_0.10	21.80 ± 2.80	23111.30 ± 1340.10
EA(SA-m)_0.20	22.00 ± 2.50	22699.50 ± 1897.40
EA(SA-m)_0.50	26.50 ± 2.50	22617.00 ± 2070.60
EA(SA-m)_0.80	32.90 ± 1.90	22545.50 ± 3958.00
EA(SA-m)_0.90	34.80 ± 2.00	19079.10 ± 7452.80
EA(SA-m)_0.95	35.76 ± 2.00	19490.70 ± 6899.00
EA(SA-m)_1.00	36.70 ± 2.00	19876.60 ± 6845.10
EA(SA-l)_0.001	31.90 ± 3.00	23707.30 ± 587.50
EA(SA-l)_0.05	17.70 ± 2.00	23187.30 ± 1394.80
EA(SA-l)_0.10	16.90 ± 2.00	22877.00 ± 1557.80
EA(SA-l)_0.20	16.70 ± 1.90	22803.50 ± 1663.90
EA(SA-l)_0.50	18.70 ± 1.50	19013.50 ± 6477.00
EA(SA-l)_0.80	19.30 ± 1.70	14709.00 ± 7251.50
EA(SA-l)_0.90	19.67 ± 1.70	14986.50 ± 7308.20
EA(SA-l)_0.95	19.60 ± 1.60	13019.30 ± 6787.90
EA(SA-l)_1.00	19.70 ± 1.60	15455.00 ± 7173.00

EA+LS

Algorithm	% Satisfactory movements	Number of evaluations
EA	-	23615.90 ± 732.20
EA+HC_10%_50	100.00 ± 0.00	23347.50 ± 55.90
EA+HC_10%_100	100.00 ± 0.00	22598.40 ± 55.90
EA+HC_10%_200	100.00 ± 0.00	21096.80 ± 56.90
EA+HC_15%_50	100.00 ± 0.00	22945.50 ± 29.60
EA+HC_15%_100	100.00 ± 0.00	21883.30 ± 23.30
EA+HC_15%_200	100.00 ± 0.00	19647.30 ± 30.20
EA+HC_20%_50	100.00 ± 0.00	22598.40 ± 55.90
EA+HC_20%_100	100.00 ± 0.00	21096.80 ± 56.90
EA+HC_20%_200	100.00 ± 0.00	18099.00 ± 57.20
EA+SA-m_10%_50	100.00 ± 0.00	23347.50 ± 55.90
EA+SA-m_10%_100	100.00 ± 0.00	22598.40 ± 55.90
EA+SA-m_10%_200	100.00 ± 0.00	21096.80 ± 56.90
EA+SA-m_15%_50	100.00 ± 0.00	22945.50 ± 29.60
EA+SA-m_15%_100	100.00 ± 0.00	21883.30 ± 23.30
EA+SA-m_15%_200	100.00 ± 0.00	19647.30 ± 30.20
EA+SA-m_20%_50	100.00 ± 0.00	22598.00 ± 55.90
EA+SA-m_20%_100	100.00 ± 0.00	21096.80 ± 56.90
EA+SA-m_20%_200	100.00 ± 0.00	18099.00 ± 57.20
EA+SA-l_10%_50	50.00 ± 1.80	23347.50 ± 55.90
EA+SA-l_10%_100	49.80 ± 1.20	22598, 40 ± 55.90
EA+SA-l_10%_200	49.90 ± 0.90	21096.80 ± 56.90
EA+SA-l_15%_50	50.03 ± 1.40	22945.50 ± 29.60
EA+SA-l_15%_100	49.82 ± 1.10	21883.30 ± 23.30
EA+SA-l_15%_200	49.93 ± 0.90	19647.30 ± 30.20
EA+SA-l_20%_50	49.90 ± 1.30	22598.40 ± 55.90
EA+SA-l_20%_100	49.90 ± 0.90	21096.80 ± 56.90
EA+SA-l_20%_200	50.00 ± 0.70	18099.00 ± 57.20

General parameters:
150 individuals
24000 evaluations
4 design problems
30 executions/conf.

Local search techniques:
HC
SA+Metropolis (m)
SA+Logistic (l)

Experiments and results

Improvement on software metrics

EA(LS)

- ✓ Most important improvements: CB, CS
- ✓ Between 1% and 6%
- ✓ For all problem instances
- ✗ Problems to maintain the trade-off between metrics

EA+LS

- ✓ Improved metric: CB
- ✗ Between 1% and 2%
- ✗ Mainly for the easiest problem instances
- ✗ Not a significant difference to the software architect

Concluding remarks

- **Experimental outcomes** indicate that:
  - Executing LS after the evolution is not effective
  - Including LS within the evolution speeds up the optimisation
  - Improvements mostly appear for those metrics controlling the size of the components
  - The baseline algorithm maintains a better trade-off

- **Future work**
  - **Domain knowledge** to guide the generation of neighbours
  - Application to a **multi-objective** problem formulation

Memetic Algorithms for the Automatic Discovery of Software Architectures

Thanks!

UNIVERSIDAD DE CÓRDOBA

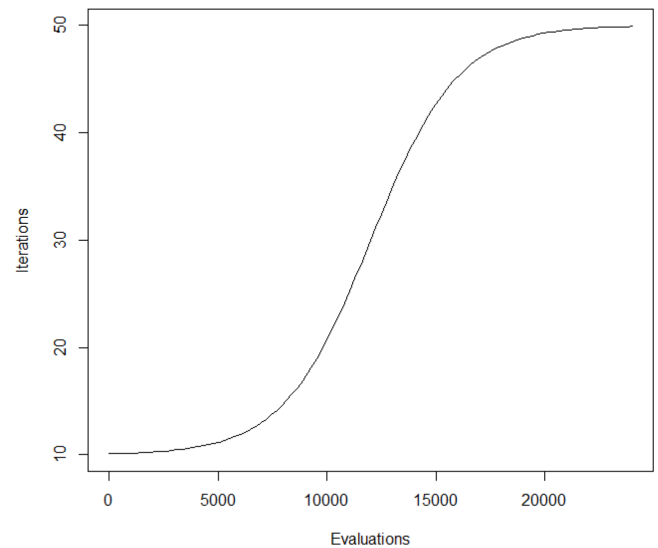
Aurora Ramírez

Email. **aramirez@uco.es**
Web. **http://www.uco.es/users/aramirez/en**

Experiments and results

Parameter study

General parameters	LS techniques	HC, SA+Exp, SA+Log
	Initial temperature (SA)	Prob. worst individual = 50% in the initial state
	Cooling factor (SA)	0.95
	Max. Evaluations	24,000
	Population size	150
	Problem instances	4 designs (32-59 classes)
EA(LS)	LS probability	0.01, 0.05, 0.1, 0.2, 0.8, 0.9, 0.95, 0.99, 1.0
	No. Iterations	Between MIN (10) and MAX (50)
EA+LS	% Solutions	10%, 15%, 20%
	No. Iterations	50, 100, 200



$$nIterLS = \frac{MAX - MIN}{1 + e^{-k\left(\frac{nEvals - \max Evals}{2}\right)}}$$