There is always room for one more, and for many more

Aurora Ramírez

Dept. Computer Science and Numerical Analysis

University of Córdoba, Spain

How many aspects should software engineers consider when developing software?

Can we optimise *many metrics* in Search Based Software Engineering?

TOO MANY!

OF COURSE!

Content

Introduction

- The importance of **measurement** in Soft. Eng.
  - Metrics appear in every phase of the software development process
  - Different perspectives of the software quality
- Metrics as **fitness functions** in SBSE
  - A common approach to evaluate candidate solutions
  - Well-established frameworks: coupling and cohesion (design), coverage (testing), time and cost(project management)...

Introduction

- **SBSE** can be considered a mature field...
  - Optimisation problems in almost every phase
  - Experimental studies, some tools and industrial experiences...
  - A world-wide community with specialised events
- ...however...
  - We mostly use *simple problem formulations* (1-3 objectives)
  - We mostly use *traditional algorithms* (e.g. NSGA-II)

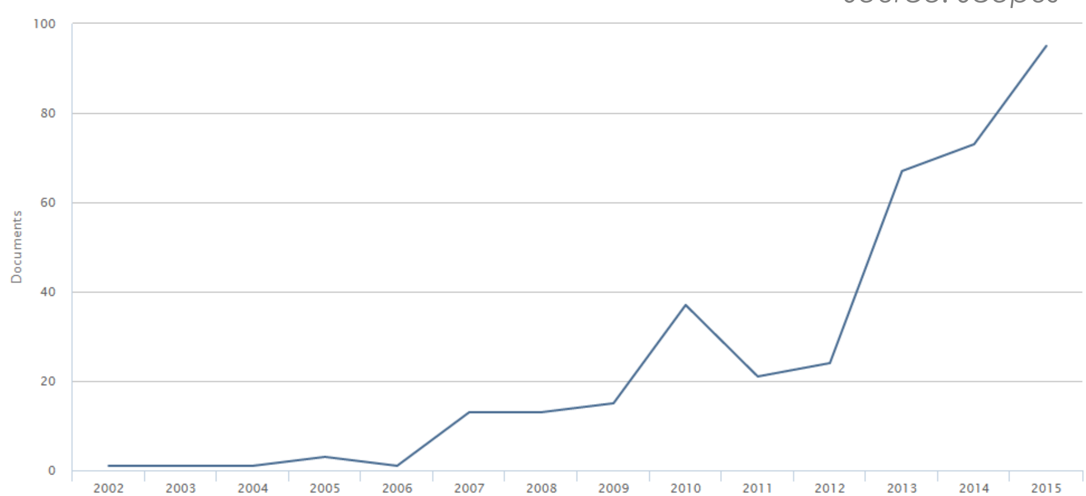
Many-objective optimisation

- **Historical view**
  - First time mentioned in *(Farina and Amato, 2002)*
  - Identification of key issues (2003-2007)
  - Proposals of algorithms, surveys... (recent years)

- **Hot-topic** in Evolutionary Computation

398 results (2002-2016)
67% of them in the last 3 years

Source: Scopus



Many-objective optimisation

- Many-objective optimisation problems (MaOPs)
  - The same definition that multi-objective problems (MOPs)

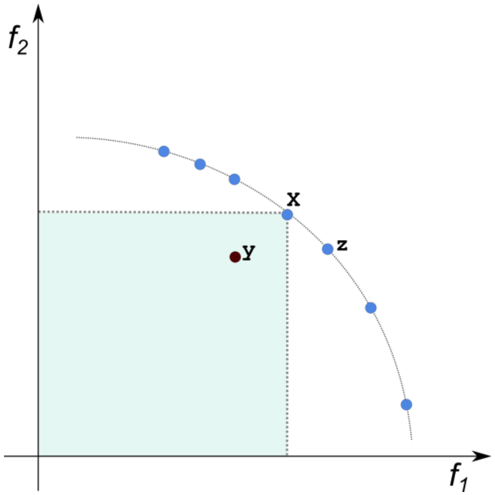
$$\max F(x) = (f_1(x), f_2(x), \ldots, f_m(x))$$
$$subject\ \ to\ x \in \Omega, \quad x = (x_1, x_2, \ldots, x_n)$$

  - At least **4 objectives** (general agreement)
  - Synthetic test problems can be defined with hundreds

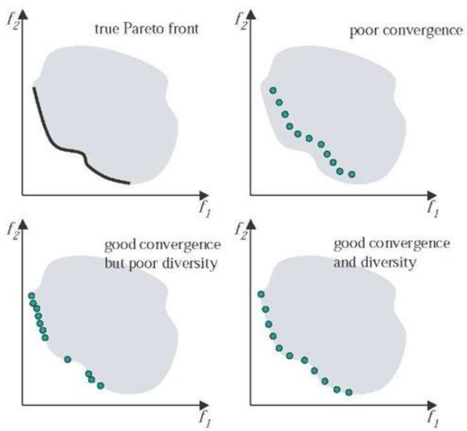
Many-objective optimisation

- The **Pareto dominance** principle

$$x, y \in \Omega, \quad x \prec y \quad iff$$
$$\forall i \in \{1,\ldots,m\},\ f_i(x) \geq f_i(y) \quad and \quad \exists j \in \{1,\ldots,m\},\ f_j(x) > f_j(y)$$



- Pareto set (PS) and Pareto front (PF)
- The **goals** are...
  - Convergence to the true Pareto front
  - Diversity of the returning solution set



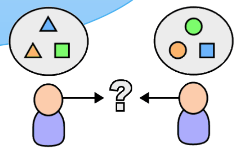
Many-objective optimisation

Main difficulties

Number of non-dominated solutions

Diversity preservation

Inefficiency of operators



Complete representation of the PF

Performance measures

Visualisation of trade-offs

Adapted from (Deb and Jain, 2014)

Many-objective optimisation

- Current approaches

(von Lücken et al., 2014)
(Li et al., 2015)

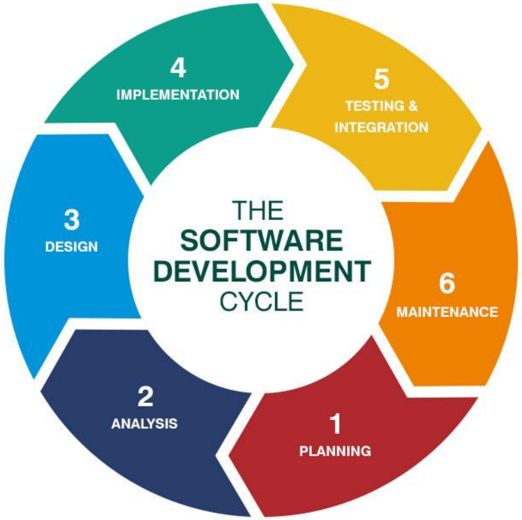
Technique	Algorithms
Relaxed dominance	ε-MOEA, GrEA, MDMOEA
Diversity techniques	NSGA-II+SDE, SPEA2+SDE
Aggregation techniques	MSOPS, MODELS, MOEA/D
Quality indicators	HypE, IBEA, SMS-EMOA
Reference set	NSGA-III, TC-SEA, TAA
Use of preferences	MQEA-PS, PICEA, SBGA
Reduction of objectives	MOSS/EMOSS, PCSEA, SIBEA

SBSE needs many-objective optimisation

“Measurement is the first step that leads to control and eventually to improvement. If you can’t measure something, you can’t understand it. If you can’t understand it, you can’t control it. If you can’t control it, you can’t improve it.”

(H. James Harrignton)

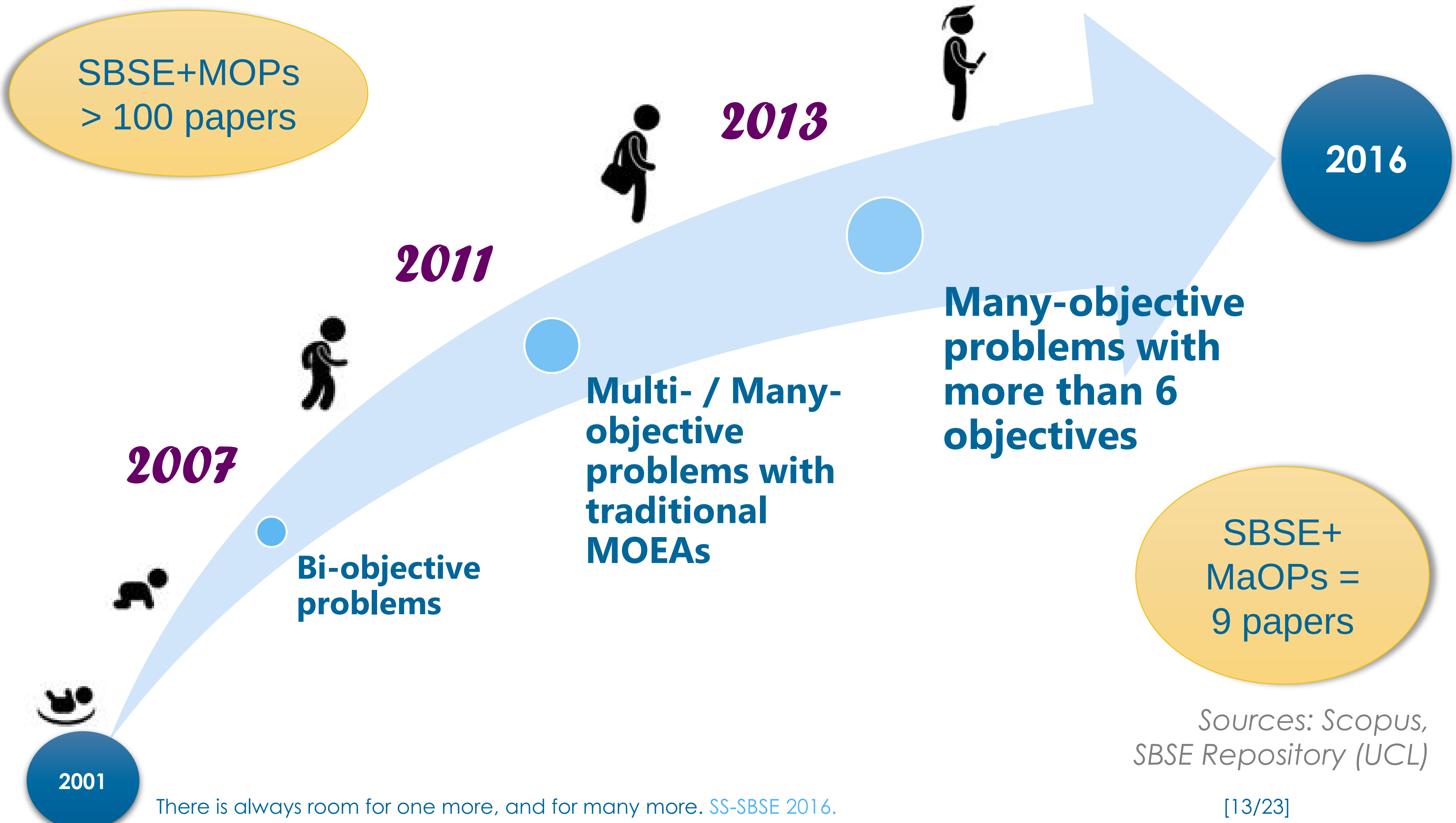
SOFTWARE ENGINEERS NEED METRICS!



SBSE needs many-objective optimisation

- Metric suites
  - (Chidamber and Kemerer, 1994): 6 metrics for OO design
  - (Bansiya and Davis, 2002): 11 metrics derived from ISO 9126
  - (Abdellatief *et al*. , 2013): review of 23 metrics for CBSS
- Software quality standards
  - ISO 9126: 6 characteristics divided into 27 subcharacteristics
  - ISO 25000 (SQuaRE): 8 characteristics and 31 subcharactecristics
- Tools
  - SDMetrics (UML diagrams): 132 metrics
  - SonarQube (code, documentation, test cases...): 77 metrics

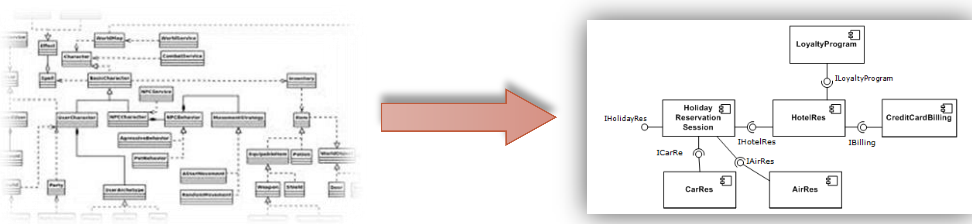
SBSE needs many-objective optimisation



Sources: Scopus, SBSE Repository (UCL)

Case study 1: discovery of software architectures

- [*SEARCH PROBLEM*] We want to identify the underlying architecture from an analysis model (class diagram)

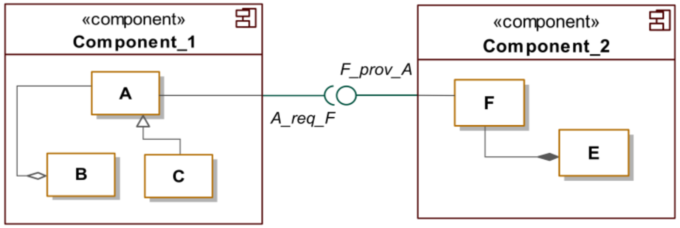


- Why we need a many-objective approach?
  - ✓ There are many metrics beyond coupling and cohesion
  - ✓ One single solution is not enough for the architect
  - ✓ Selecting and combining software metrics can be difficult

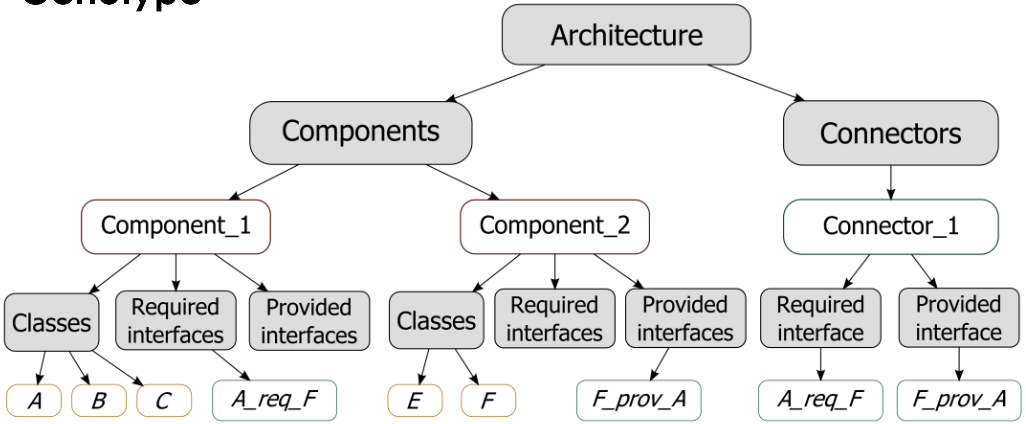
A. Ramírez, J.R. Romero, S. Ventura. “A comparative study of many-objective evolutionary algorithms for the discovery of software architectures”. *Empirical Software Engineering*. 2015. *In press*.

Case study 1: discovery of software architectures

Phenotype



Genotype



Genetic operator

- A roulette-based mutation operator:
  - *Add a component*
  - *Remove a component*
  - *Merge two components*
  - *Split a component*
  - *Move a class*

Initialisation and constraints

1. Random distribution of classes
   - ✓ No empty components and no replicated classes
2. Assignment of interfaces to components and connectors
   - ✗ Isolated or mutually dependant components

Case study 1: discovery of software architectures

- One of the most important quality criteria for component-based architectures is **maintainability** (ISO Std. 25000):
  - ✓ **Modularity**. A change to one component has a minimal effect on others
  - ✓ **Reusability**. An asset can be used in more than one solution
  - ✓ **Analysability**. Parts of the software to be modified can be identified

Metric	Min/Max	Quality attribute	Range of values	Design goals
icd	max	modularity	[0, 1]	Small components with high cohesion
erp	min	modularity	[0, $*$]	Large components with low coupling
ins	min	modularity	[0, 1]	Components with few interactions
enc	max	modularity	[0, 1]	Components with hidden classes
cs	min	modularity	[0, n]	Small or medium-sized components
cl	min	modularity	[0, n]	Components with few provided interfaces
gcr	min	reusability	[1, $*$]	Connected classes within each component
abs	max	reusability	[0, 1]	Components with abstract classes
cb	max	analisability	[0, 1]	Equal-sized components

Case study 1: discovery of software architectures

$$icd_i = \frac{\#classes_{total} - \#classes_i}{\#classes_{total}} \cdot \frac{ci_i^{in}}{ci_i^{in} + ci_i^{out}} \qquad icd = \frac{1}{n} \cdot \sum_{i=1}^{n} icd_i$$

$$erp = \sum_{i=1}^{n} \sum_{j=i+1}^{n} (w_{as} \cdot n_{as_{ij}} + w_{ag} \cdot n_{ag_{ij}} + w_{co} \cdot n_{co_{ij}} + w_{ge} \cdot n_{ge_{ij}})$$

$$ins_i = \frac{ec_i}{ec_i + ac_i} \qquad ins = \frac{1}{n} \cdot \sum_{i=1}^{n} ins_i$$

$$enc_i = \frac{\#inner_{classes}}{\#total_{classes}} \qquad enc = \frac{1}{n} \cdot \sum_{i=1}^{n} enc_i$$

$$cc_{size}^{i} = \begin{cases} 1 & \text{if size}(i) > \text{ threshold} \\ 0 & \text{otherwise} \end{cases} \qquad cs = \sum_{i=1}^{n} cc_{size}^{i}$$

$$cc_{link}^{i} = \begin{cases} 1 & \text{if } \#provided\,interfaces_i > \text{ threshold} \\ 0 & \text{otherwise} \end{cases} \qquad cl = \sum_{i=1}^{n} cc_{link}^{i}$$

$$gcr = \frac{\#cgroups}{\#components}$$

$$abs_i = \frac{\#abstract\_classes_i}{\#classes_i} \qquad abs = \frac{1}{n} \cdot \sum_{i=1}^{n} abs_i$$

$$sb(n) = \begin{cases} \frac{n-1}{\mu-1} & \text{if } n < \mu \\ 1 - \frac{n-\mu}{\omega-\mu} & \text{if } \mu < n < \omega \\ 0 & \text{if } n \geq \omega \end{cases}$$

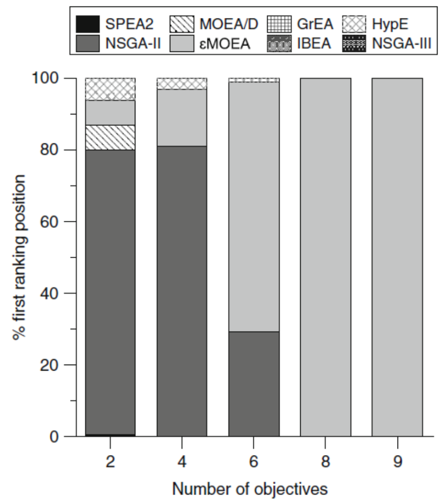
$$csu(C) = 1 - Gini(\{volume(c) : c \in C\})$$

$$cb(S) = sb(|C|) \cdot csu(C)$$

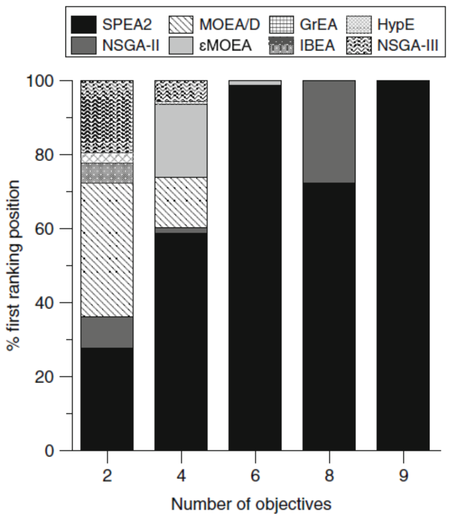
Case study 1: discovery of software architectures

- From the **evolutionary perspective**...
  - ✓ For 2- and 4-objective problems:
    - MOEAs are valid algorithms ... as expected!
    - **NSGA-II** overcomes to the rest of algorithms
    - SPEA2 and MOEA/D provide good spread of solutions
  - ✓ For more than 6 objectives:
    - Not all the algorithms behave the same, or scale similarly
    - **ε-MOEA and HypE** apparently overcome now
    - NSGA-II is still competitive
    - NSGA-III disappoints the expectations

- **BUT** ... the evolutionary perspective **may not match** the software architect's perspective!



(a) Hypervolume



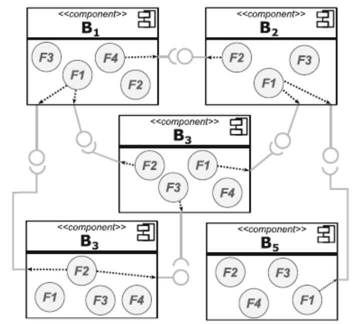
(b) Spacing

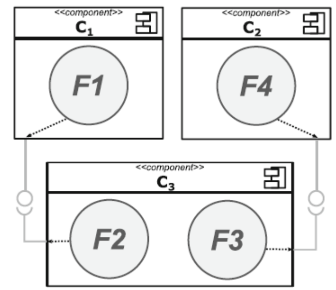
Case study 1: discovery of software architectures

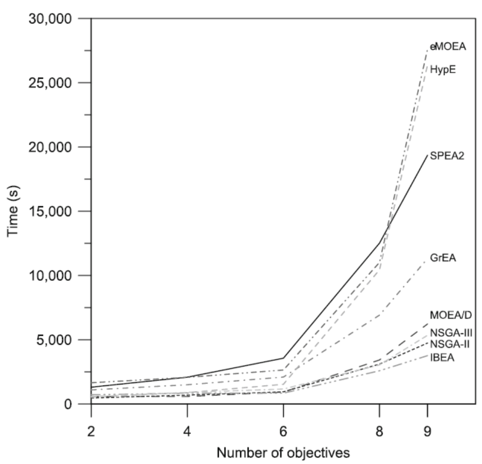
- From the **architect's perspective**, we need to keep in mind that:

Time may hamper its applicability to decision-support tools

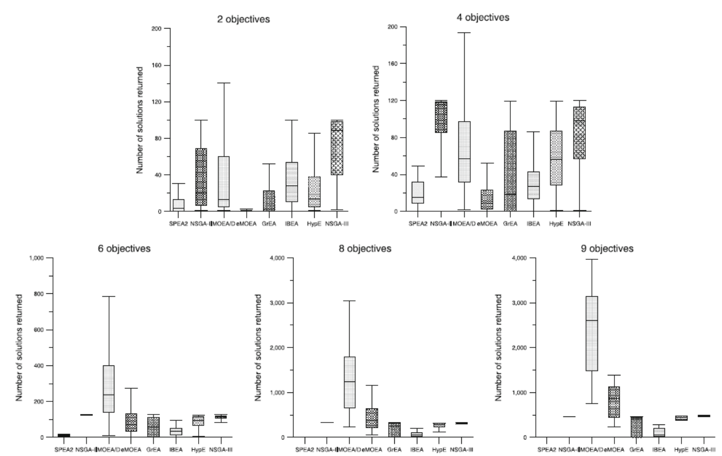
The selected metrics greatly influence the type of architectural solutions







The **number of solutions** returned depends on the number of metrics and the selected algorithm



Case study 2: QoS-aware composition of web services

A well-known and studied optimisation problem in Service Oriented Computing

A candidate solution represents a possible assignment of concrete services to abstract tasks defining a structure of composition

Find the solutions that maximise the global Quality of Service (QoS): cost, latency...

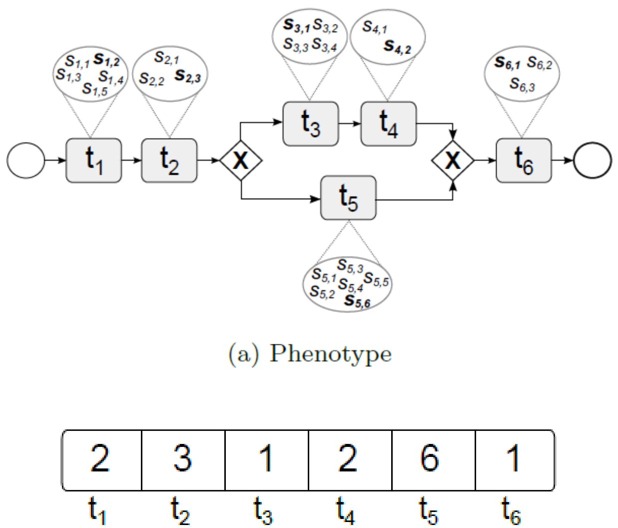
Existing SBSE approaches

- Metaheuristic techniques
  - Evolutionary algorithms (**MOEAs**)
  - GRASP with Path Relinking
  - Particle Swarm Optimisation
  - ...
- Problem formulation
  - Single-objective (aggregation)
  - Multi-objective (**5-10** QoS properties)

Case study 2: QoS-aware composition of web services

The 9 QoS properties

1. Response Time
2. Availability
3. Reliability
4. Throughput
5. Latency
6. Successability
7. Compliance
8. Best practices
9. Documentation



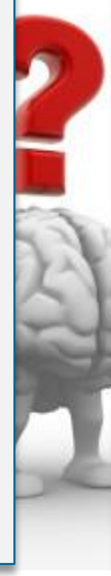
(a) Phenotype

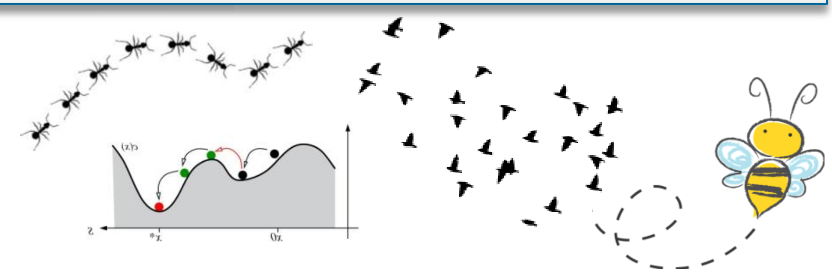
(b) Genotype

QoS values from 2507 real-world web services

Open issues

SOFTWARE METRICS	SEARCH ALGORITHMS
✓ Study of **available** metrics	✓ New **algorithms** in many-objective optimisation
✓ Definitions based on quality models and **standards**	✓ Adequacy of the **families** of algorithms to SBSE problems
✓ Quality attributes as objective functions	✓ Other **metaheuristics** (ACO, LS)
✓ **Dependencies** between metrics	✓ **Specific** developments for SBSE





Conclusions

Search Based Software Engineering can benefit from the ongoing advances in many-objective optimisation

- From the point of view of **SBSE**
  - SBSE requires more sophisticated methods
  - Experimental studies to assess the performance
- From the point of view of **many-objective optimisation**
  - SBSE might be a source of complex MaOPs
  - New techniques beyond evolutionary computation

There is always room for one more, and for many more

UNIVERSIDAD DE CÓRDOBA

Aurora Ramírez

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

Thank you!

References

Abdellatief, M., Md Sultan, A.B., Ghani, A.A.A., Jabar, M.A. “A mapping study to investigate component-based software system metrics”. *The Journal of Systems and Software*, vol. 86, pp- 587-603. 2013.

Adra, S., Fleming, P. “Diversity management in evolutionary many-objective optimization”. *IEEE Transactions on Evolutionary Computation*, vol. 15(2), pp. 183-195. 2011.

Aleti, A., Buhnova, B., Grunske, L., Koziolek, A., Meedeniya, I. “Software Architecture Optimization Methods: A Systematic Literature Review”, *IEEE Transactions on Software Engineering*, vol. 39(5), pp. 658-683. 2013.

Bansiya, J., Davis, C-G. “A hierarchical model for object-oriented design quality assessment”. *IEEE Transactions on Software Engineering*, vol. 28(1), pp. 4-17.

Chand, S., Wagner, M. “Evolutionary many-objective optimization: A quick-start guide”. *Surveys in Operational Research and Management Science*, vol. 20, pp.35-42. 2015.

Chidamber., S.R., Kemerer, C.F. “A Metrics Suite for Object Oriented Design”. *IEEE Transactions on Software Engineering*, vol. 20(6), pp.476-493. 1994.

Deb, K., Jain, H. “An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point-Based Nondominated Approach, Part I: Solving Problems With Box Constraints”. *IEEE Transactions on Evolutionary Computation*, vol. 18(4), pp. 577-601. 2014.

Farina, M., Amato, P. “On the optimal solution definition for many-criteria optimization problems”. *Annual Meeting of the North American Fuzzy Information Processing Society (NAFIPS)*, pp.233-238 2002.

Harman, M., Afshin Mansouri, S., Zang, Y. “Search Based Software Engineering: Trends, Techniques and Applications”. *ACM Computing Surveys*, vol. 45(1), Article 11, 2012.

References

Kalboussi, S., Cehikh, S., Kessentini, M., Ben Said, L. “Preference-Based Many-Objective Evolutionary Testing Generates Harder Test Cases for Autonomous Agents”. *Proc. International Symposium on Search Based Software Engineering (SSBSE)*, pp. 245-250. 2013.

Li, B., Li, J., Tang, K.., Yao, X. “Many-Objective Evolutionary Algorithms: A Survey”. *ACM Computing Surveys*, vol. 48(1), Article 13. 2015.

Mkaouer, W., Kessentini M., Deb, K., Ó Cinnéide, M. “High Dimensional Search-based Software Engineering: Finding Tradeoffs among 15 Objectives for Automating Software Refactoring using NSGA-III”. *Proc. Annual Conference on Genetic and Evolutionary Computation (GECCO’14)*, pp. 1263-1270. 2014.

Mkaouer, W., Kessentini, M., Kontchou, P., Deb, K., Bechikh, S., Ouni, A.. “Many-Objective Software Remodularization using NSGA-III”. *ACM Transactions on Software Engineering and Methodology*, vol 24(3), No. 17. 2015.

Mkaouer, M.W., Kessentini, M., Bechikh, S., Ó Cinneide, M., Deb, K. “On the use of many quality attributes for software refactoring: a many-objective search-based software engineering approach”. *Empirical Software Engineering*. 2015. *In press*.

Panichella, A., Kifetew, F.M., Tonella, P. “Reformulating Branch Coverage as a Many-Objective Optimization Problem”. *Proc. IEEE 8th International Conference on Software Testing, Verification and Validation (ICST)*, pp. 1-10. 2015.

Parejo, J.A., Segura, S., Fernández, P. Ruiz-Cortés, A. “QoS-aware web services composition using GRASP with Path Relinking”. Expert Systems with Applications, vol. 41(9), pp. 4211-4223. 2014.

References

Praditwong, K., Yao, X. “How well do multi-objective evolutionary algorithms scale to large problems”. *Proc. IEEE Congress on Evolutionary Computation (CEC’07)*, pp. 3959-3966. 2007.

Purshouse, R.C., Fleming, P.J. “Evolutionary many-objective optimisation: an exploratory analysis”. *IEEE Congress on Evolutionary Computation (CEC’03)*, vol. 3, pp. 2066-2073. 2003.

Purshouse, R., Fleming, P. “On the evolutionary optimization of many conflicting objectives”. *IEEE Transactions on Evolutionary Computation*, vol. 11(6), pp. 770-784. 2007.

Ramírez, A., Romero, J.R., Ventura, S. “On the Performance of Multiple Objective Evolutionary Algoorithms for Software Architecture Discovery”. *Proc. Annual Conference on Genetic and Evolutionary Computation (GECCO’14)*, pp. 1287-1294.

Ramírez, A., Romero, J.R., Ventura, S. “A comparative study of many-objective evolutionary algorithms for the discovery of software architectures”. *Empirical Software Engineering*. 2015. *In press*.

Sayyad, A.S., Ammar, H. “Pareto-Optimal Search-Based Software Engineering (POSBSE): A Literature Survey”. *2nd International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering (RAISE)*, pp. 21-27. 2013.

Schutze, O. Lara, A., Coello Coello, C.A. “On the influence of the number of objectives on the hardness of a multiobjective optimization problem”. *IEEE Transactions on Evolutionary Computation*, vol. 15(4), pp. 444-455.

von Lücken, C., Barán, B., Brizuela, C. “A survey on multi-objective evolutionary algorithms for many-objective problems”. *Computational Optimization and Applications*, vol. 58(3), pp. 707-756. 2014.

Walker, D.J., Everson, R.M., Fieldsend, J.E. “Visualizing Mutually Nondominating Solution Sets in Many-Objective Optimization”. *IEEE Transactions on Evolutionary Computation*, vol. 17(2), pp. 165-184. 2013.