Advances in Engineering Software 42 (2011) 760–771

Contents lists available at ScienceDirect

Advances in Engineering Software

journal homepage: www.elsevier.com/locate/advengsoft

jMetal: A Java framework for multi-objective optimization

Juan J. Durillo, Antonio J. Nebro *

Departamento de Lenguajes y Ciencias de la Computación, University of Málaga, Spain

ARTICLE INFO

Article history:
Received 30 November 2009
Received in revised form 19 November 2010
Accepted 16 May 2011
Available online 12 June 2011

Keywords:
Multi-objective optimization
Metaheuristics
Software tool
Object-oriented architecture
Performance assessment support
Experimentation

ABSTRACT

This paper describes jMetal, an object-oriented Java-based framework aimed at the development, experimentation, and study of metaheuristics for solving multi-objective optimization problems. jMetal includes a number of classic and modern state-of-the-art optimizers, a wide set of benchmark problems, and a set of well-known quality indicators to assess the performance of the algorithms. The framework also provides support to carry out full experimental studies, which can be configured and executed by using jMetal's graphical interface. Other features include the automatic generation of statistical information of the obtained results, and taking advantage of the current availability of multi-core processors to speed-up the running time of the experiments. In this work, we include two case studies to illustrate the use of jMetal in both solving a problem with a metaheuristic and designing and performing an experimental study.

1. Introduction

Most of the optimization problems in the real world are multi-objective in nature, which means that solving them requires to optimize two or more contradictory functions or objectives. These problems are known as multi-objective optimization problems (MOPs). The searched optimum of this kind of problems is not a single solution as in the mono-objective case, but a set of solutions known as the Pareto optimal set. Any element in this set is no better than the other ones for all the objectives. This set is given to the decision maker, who has to choose the best trade-off solution according to his/her preferences. As whatever optimization problem, MOPs may present features such as epistasis, deceptiveness, or NP-hard complexity [1], making them difficult to solve. Furthermore, if we consider engineering problems, we find frequently that some of the functions composing them are non-linear and they can be computationally expensive to evaluate.

In those situations, exact techniques are often not applicable, so approximated methods are mandatory. Like in mono-objective optimization, metaheuristics [2,3] are approximated algorithms for solving MOPs. Among them, evolutionary algorithms are very popular [4,5], and some of the most well-known algorithms in this field belong to this class (e.g. NSGA-II [6], PAES [7], SPEA2 [8]). Nevertheless, other metaheuristic techniques are also gaining momentum, as particle swarm optimization [9].

Multi-objective optimization using metaheuristics is an active research field where new techniques are continuously being proposed to cope with real settings (such as uncertainty and noise, many-objective optimization, and convergence speed). In this context, the use of software frameworks is a valuable tool for helping in the implementation of new ideas, facilitated by reusing code of existing algorithms, and for understanding the behavior of existing techniques. Desirable features of these kinds of software tools comprise:

- to include state-of-the-art algorithms,
- to contain commonly accepted benchmark MOPs,
- to provide quality indicators for performance assessment, and
- to assist users in carrying out research studies.

Some years ago, it was difficult to find any software package satisfying those requirements. The implementation in C of NSGA-II, the most used multi-objective metaheuristic algorithm, was publicly available,[1] but it was difficult to be used as the basis of new algorithms, in part due to its lack of an object-oriented design. An interesting choice was (and still is) PISA [10], a C-based framework for multi-objective optimization which is based on separating the algorithm specific part of an optimizer from the application-specific part. This is carried out using a shared-file mechanism to communicate the module executing the application with the module running the metaheuristic. However, a drawback of PISA is that their internal design hinders to reuse code.

Those reasons leaded us to develop a new software toolbox starting from scratch, and the result was jMetal, a Java-based framework designed to multi-objective optimization using metaheuristics. We imposed ourselves as design goals that jMetal

* Corresponding author.
E-mail addresses: durillo@lcc.uma.es (J.J. Durillo), antonio@lcc.uma.es (A.J. Nebro).

[1] NSGA-II: http://www.iitk.ac.in/kangal/codes.shtml.

0965-9978/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.advengsoft.2011.05.014

should be simple and easy to use, portable (hence the choice of Java), flexible, and extensible [11]. Nowadays, jMetal is publicly available to the community of people interested in multi-objective optimization. It is licensed under the GNU Lesser General Public License,[2] and it can be obtained freely from http://jmetal.sourceforge.net.

During the development of jMetal, other Java-based software tools have emerged, e.g., EVA2,[3] OPT4j,[4] or ECJ.[5] Other similar tools based in other programming languages have also come into light, as MOEAT [12], that is implemented in C#, what restricts them only to Windows-based PCs. All these toolboxes can be useful enough for many researchers. However, while jMetal is specifically oriented to multi-objective optimization with metaheuristics, most of those frameworks are mainly focused on evolutionary algorithms, and many of them are centered in single-objective optimization, offering only extensions to the multi-objective domain.

jMetal has the following features that, all together, make our framework a unique system compared to existing proposals:

- Implementation of a number of modern multi-objective optimization algorithms: NSGA-II [6], SPEA2 [8], PAES [7], PESA-II [13], OMOPSO [14], MOCell [15], AbYSS [16], MOEA/D [17], Densea [18], CellDE [19], GDE3 [20], FastPGA [21], IBEA [22], SMPSO [23], MOCHC [24], and SMS-EMOA [25].
- Validation of the implementation: we compared our implementations of NSGA-II and SPEA2 with the original versions, achieving competitive results [11].
- A rich set of test problems including:
  - Problem families: Zitzler–Deb–Thiele (ZDT) [26], Deb–Thiele–Laumanns–Zitzler (DTLZ) [27], Walking-Fish-Group (WFG) test problems [28]), CEC2009 (unconstrained problems) [29], and the Li-Zhang benchmark [17].
  - Classical problems: Kursawe [30], Fonseca and Flemming [31], Schaffer [32].
  - Constrained problems: Srinivas [33], Tanaka [34], Osyczka2 [35], Constr_Ex [6], Golinski [36], Water [37].
- Implementation of the most widely used quality indicators: Hypervolume [38], Spread [6], Generational Distance [39], Inverted Generational Distance [39], Epsilon [40].
- Different variable representations: binary, real, binary-coded real, integer, permutation.
- Support for performing experimental studies, including the automatic generation of LaTeX tables with the results after applying quality indicators, statistical pairwise comparison by using the Wilcoxon test to the obtained results, and R (http://www.r-project.org/) boxplots summarizing those results. In addition, jMetal includes the possibility of using several threads for performing these kinds of experiments in such a way that several independent runs can be executed in parallel using modern multi-core CPUs.
- A Graphical User Interface (GUI) for giving support in solving problems and performing experimental studies.
- A Web site (http://jmetal.sourceforge.net) containing the source codes, the user manual and, among other information, the Pareto fronts of the included MOPs, references to the implemented algorithms, and references to papers using jMetal.

We have used jMetal in a number of works: we have proposed new multi-objective techniques (AbYSS [16], a scatter search algorithm; SMPSO, a particle swarm optimization technique [23]; or CellDE [19], a cellular genetic algorithm hybridized with differential evolution) and we have investigated properties of state-of-the-art multi-objective optimizers (convergence speed [41], behavior when solving parameter scalable problems [42,43], or the influence of using a steady-state selection scheme in multi-objective genetic algorithms [44]). Besides our own work, jMetal has been used by other groups, as it can be seen in the publication section of jMetal's Web site.

In this paper, our purpose is to describe jMetal (version 3.1) and how it can be used by people interested in using metaheuristics for solving MOPs. Since many researches in the multi-objective field are not only interested in solving a given problem but also in assessing the performance of different algorithms and comparing them when solving a given benchmark, we also describe here how to use our tool in that sense. In particular, to cope with this last issue, we have considered a case study consisting in solving a benchmark composed of four constrained problems (Osyczka2, Srinivas, Golinski, and Tanaka) by using four different multi-objective algorithms (NSGA-II, SPEA2, MOCell and AbYSS).

The content of this paper can be summarized as follows. In the next section we include some background on multi-objective optimization. Section 3 is aimed at presenting the jMetal architecture and its main components. The quality indicators included in jMetal are described in Section 4. We describe in Section 5 how to extend the framework with a new problem and solving it. After that, we explain hot to develop an algorithm using jMetal. Section 7 is devoted to describing the case study and how the framework can be used to carry out a full comparison of experiments. Finally, Section 8 presents the conclusions and further work.

2. Multi-objective background

In this section, we include some background on multi-objective optimization. We define formally the concept of MOP, Pareto optimality, Pareto dominance, Pareto optimal set, and Pareto front. In these definitions we are assuming, without loss of generality, that minimization is the goal for all the objectives. A general MOP can be formally defined as follows:

Definition 1 (*MOP*). Find a vector $\vec{x}^* = [x_1^*, x_2^*, \ldots, x_n^*]$ which satisfies the m inequality constraints $g_i(\vec{x}) \geqslant 0, i = 1, 2, \ldots, m$, the p equality constraints $h_i(\vec{x}) = 0,\ i = 1, 2, \ldots, p$, and minimizes the vector function $\vec{f}(\vec{x}) = [f_1(\vec{x}), f_2(\vec{x}), \ldots, f_k(\vec{x})]^T$, where $\vec{x} = [x_1, x_2, \ldots, x_n]^T$ is the vector of decision variables.

The set of all values satisfying the constraints defines the *feasible region* Ω and any point $\vec{x} \in \Omega$ is a *feasible solution*. As mentioned before, we seek the *Pareto optima*. More formally:

Definition 2 (*Pareto optimality*). A point $\vec{x}^* \in \Omega$ is Pareto Optimal if for every $\vec{x} \in \Omega$ and $I = \{1,2,\ldots,k\}$ either $\forall_{i \in I}(f_i(\vec{x}) = f_i(\vec{x}^*))$ or there is at least one $i \in I$ such that $f_i(\vec{x}) > f_i(\vec{x}^*)$.

This definition states that $\vec{x}^*$ is Pareto optimal if no other feasible vector $\vec{x}$ exists which would improve some criteria without causing a simultaneous worsening in at least one other criterion. Other important nomenclature associated with Pareto optimality are defined below:

Definition 3 (*Pareto dominance*). A vector $\vec{u} = (u_1, \ldots, u_k)$ is said to dominate $\vec{v} = (v_1, \ldots, v_k)$ (denoted by $\vec{u} \preccurlyeq \vec{v}$) if and only if $\vec{u}$ is partially less than $\vec{v}$, i.e., $\forall i \in \{1,\ldots,k\}, u_i \leqslant v_i \wedge\ \exists i \in \{1,\ldots,k\}: u_i < v_i$.

Definition 4 (*Pareto optimal set*). For a given MOP $\vec{f}(\vec{x})$, the Pareto optimal set is defined as $\mathcal{P}^* = \{\vec{x} \in \Omega | \neg \exists \vec{x}' \in \Omega, \vec{f}(\vec{x}') \preccurlyeq \vec{f}(\vec{x})\}$.

Definition 5 (*Pareto front*). For a given MOP $\vec{f}(\vec{x})$ and its Pareto optimal set $\mathcal{P}^*$, the Pareto front is defined as $\mathcal{PF}^* = \{\vec{f}(\vec{x}), \vec{x} \in \mathcal{P}^*\}$.

[2] LGPL License: http://creativecommons.org/licenses/LGPL/2.1/.
[3] EVA2: http://www.ra.cs.uni-tuebingen.de/software/EvA2/.
[4] OPT4j: http://opt4j.sourceforge.net/.
[5] ECJ: http://cs.gmu.edu/eclab/projects/ecj/.