INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

ParadisEO-MOEO: A Framework to Design Metaheuristics for Multi-objective Optimization

Arnaud Liefooghe — Thomas Legrand — Laetitia Jourdan — El-Ghazali Talbi

N° 207

July 2007

Thème NUM

ISSN 0249-6399 ISRN INRIA/RR--207--FR+ENG

ParadisEO-MOEO: A Framework to Design Metaheuristics for Multi-objective Optimization

Arnaud Liefooghe, Thomas Legrand, Laetitia Jourdan, El-Ghazali Talbi

Thème NUM — Systèmes numériques
Projet Dolphin

Rapport de recherche n 207 — July 2007 — 18 pages

Abstract: This paper presents ParadisEO-MOEO, a white-box object-oriented generic framework dedicated to the flexible design of metaheursitics for multi-objective optimization. This paradigm-free software embeds some features and techniques for scalar and Pareto-based resolution and aims to provide a set of classes allowing to ease and speed up the development of computationally efficient programs. It is based on a clear conceptual distinction between the solution methods and the multi-objective problems they are intended to solve. This separation confers a maximum design and code reuse. ParadisEO-MOEO provides a broad range of fitness assignment strategies (the achievement functions, the schemes used in NSGA, IBEA and more), the most common diversity preservation mechanisms (sharing, crowding), a broad range of elitist-related features as well as some statistical tools.

Key-words: Metaheuristics, multi-objective optimization, flexible design, code reuse, object-oriented library.

Unité de recherche INRIA Futurs
Parc Club Orsay Université, ZAC des Vignes,
4, rue Jacques Monod, 91893 ORSAY Cedex (France)
Téléphone : +33 1 72 92 59 00 — Télécopie : +33 1 60 19 66 08

1 Introduction

Nowadays, the usefulness of Multi-Objective Optimization (MOO) is globally established in the whole operational research community. Furthermore, evolutionary algorithms (EAs) are commonly used to solve multi-criterion problems since they naturally found a well-diversified set of good-quality solutions. EAs [11] are stochastic optimization processes based on an iterative improvement of a population of solutions (called individuals). Several frameworks such as MOEA [22], MOMHLib++, Open BEAGLE [8], PISA [2], TEA [6] (to only quote them) already attempt to simplify and accelerate the development process of evolutionary MOO applications. We here propose a new library, called ParadisEO-MOEO, that aims to produce efficient programs while having a minimal programming effort and a maximum code reuse. ParadisEO-MOEO is an extension of the Evolving Objects framework [16]. It includes a broad range of reusable features and techniques related to Pareto-based MOO such as performance metrics, elitism, fitness sharing and the most common Pareto-based fitness assignment schemes. The fine-grained components of ParadisEO-MOEO confer a high genericity, flexibility, adaptability and extensibility. Thus, a genuine conceptual effort has been done in order to allow the user to write only the minimum problem-specific code and to incrementally adapt an algorithm rather than entirely re-implementing it. Moreover, ParadisEO-MOEO has already been used to solve various academic problems likewise real-world applications.

The remainder of the paper is organized as follows. Section 2 gives the necessary background about MOO. Section 3 describes the aims, the implementation and the provided features of the ParadisEO-MOEO framework. In section 4, we survey some existing applications designed using ParadisEO-MOEO Finally, the last section concludes the paper and highlights several perspectives about this work.

2 Multi-objective Optimization

Many areas of the industry (telecommunication, transport, aeronautics, chemistry, mechanical, environment, etc.) are concerned with multi-objective optimization. This field finds its roots in the 19th century with the economists Edgeworth and Pareto [20]. In this section, we briefly present some basic notions about multi-objective optimization. The reader is referred to [3, 4, 19] for more details.

A *multi-objective optimization problem* (MOP) is defined by a decision space X, an objective space Y, and $n \geq 2$ objective functions $f_1, f_2, \ldots, f_n$. Each objective function can be either minimized or maximized. Without loss of generality, we here assume than all n objective functions have to be minimized. A solution $x = (x_1, x_2, \ldots, x_k)$ is represented by a vector of k decision variables. To each solution $x \in X$ is assigned exactly one objective vector $y \in Y$ on the basis of a vector function $F : X \rightarrow Y$ with $y = F(x) = (f_1(x), f_2(x), \ldots, f_n(x))$. In general, optimization problems can be divided onto two categories: continuous optimization problems and combinatorial optimization problems. In case of a *multi-objective combinato-*

rial optimization problem (MCOP), note that the decision vector x have a finite number of possible values.

For single-objective optimization problems, a solution $x_a \in X$ is said to be better than a solution $x_b \in X$ if $f(x_a) < f(x_b)$. In the multi-objective case, comparing two solutions is much more complicated. To do so, the well-known concept of *Pareto dominance* is usually used to define a partial order amoung solutions:

Definition 1 *A solution $x_a \in X$* dominates *a solution $x_b \in X$ ($x_a \succ x_b$) if and only if $\forall i \in [1..n]$, $f_i(x_a) \leq f_i(x_b)$ and $\exists i \in [1..n]$ such as $f_i(x_a) < f_i(x_b)$.*

Once is defined the concept of Pareto dominance, we can define the concept of optimal solution, known as *Pareto optimal*, *non-dominated*, *acceptable* or *effective* solution:

Definition 2 *A solution $x^* \in X$ is* Pareto optimal *if and only if there does not exist another solution $x \in X$ such as x dominates x^*.*

The set of Pareto optimal solutions is generally denoted as *Pareto optimal set* or *Pareto front*.

3 Implementation

The implementation of ParadisEO-MOEO is conceptually divided into fine-grained components. On each level of its architecture, a set of abstract classes is proposed and a wide range of concrete classes are already provided. But, as the framework aims to be extensible, flexible and easily adaptable, all its components are generic in order to provide a modular architecture allowing to quickly and conveniently develop any new scheme with a minimum code writing. The goal is to follow new strategies coming from the litterature and, if need be, to provide any additional components required for their implementation. Also, ParadisEO-MOEO constantly evolves and new features might be added to the framework regularly in order to provide a wide range of efficient and modern concepts and to reflect the most recent academic advances in the MOO field.

This section gives a detailed description of the MOO-dependent components provided within the MOEO module of ParadisEO to design a whole efficient multi-objective solver.

3.1 Common Components

3.1.1 Representation

As seen in section 2, a solution of an optimization problem needs to be represented both in the decision space and in the objective space (respectively called *genotypic* and *phenotypic* representation while designing evolutionary algorithms). In the mono-objective case, a single value is usually used for the representation in the (uni-dimensional) objective space. For MOPs, where the criterion space is multi-dimensional, a tuple of n values (called *objective vector*) might be used for such a representation.

To do so using ParadisEO-MOEO, the first thing is to set the number of objectives for the problem under consideration and, for each one, if it has to be minimized or maximized. This can be done using the `moeoObjectiveVectorTraits` static class. Then, a class templatized with the last one and inheriting of `moeoObjectiveVector` have to be created for the modelization of an objective vector, as illustrated in figure 1. Besides, as a big majority of MOPs deals with real-coded objective values, a class modeling real-coded objective vectors is already proposed within ParadisEO-MOEO.

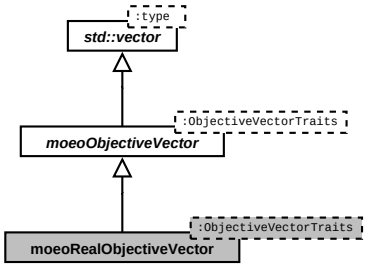

Figure 1: Core classes for the representation of a solution in the objective space.

As noticed earlier in the paper, an important issue about the design of metaheuristics for MOO is the concept of convergence and diversity. Then, we choose to give the oportunity to assign to every solution a *fitness*[1] value representing its quality in term of converence and a *diversity* value representing its quality in term of diversity (in the decision space and/or in the objective space).

Then, the class used to represent a whole solution within ParadisEO-MOEO is templatized within an objective vector type, a fitness type and a diversity type and must define its representation in the decision space, what fully depends of the tackled problem. In the implementation-level, the way to do so is to extend the `MOEO` class in order to be used for a specific problem. This modeling tends to be applicable for every kind of problem with the aim of being as general as possible. But ParadisEO-MOEO also provides easy-to-use classes for standard vector-based representations and, in particular, implementations for vectors composed of bits or of real-coded values that can thus directly be used in a ParadisEO-MOEO-designed application. This classes are summarized in figure 2 using the UML notation. Some standard variation mechanisms are provided within EO [16] and can thus directly be used for such representations.

3.1.2 Fitness Assignment Schemes

There exists a huge number of different ways to assign fitness values to a set of solutions. Following the taxonomy introduced in section 2, the fitness assignment schemes are classified onto four main categories within ParadisEO-MOEO, as illustrated in figure 3:

[1] In the multi-objective case, the word 'fitness' is not employed on the same way than it is in the mono-objective case (and in EO [16]), where it usually denotes the (single) objective value of a solution.

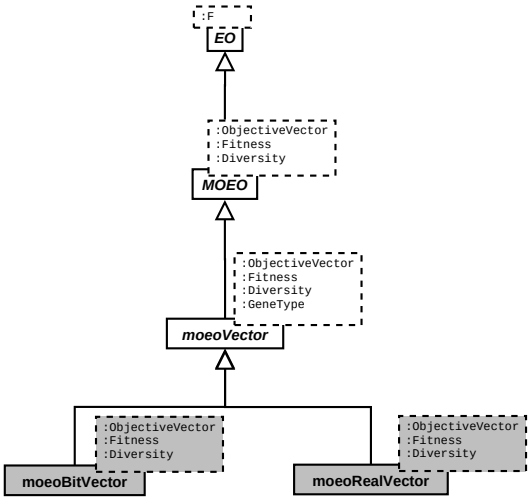

Figure 2: Core classes for the representation of a solution.

Scalar approaches: `moeoScalarFitnessAssignment`

Criterion-based approaches: `moeoCriterionBasedFitnessAssignment`

Pareto-based approaches: `moeoParetoBasedFitnessAssignment`

Indicator-based approaches: `moeoIndicatorBasedFitnessAssignment`

The details of the concrete features of ParadisEO-MOEO for fitness assignment schemes are listed below. Moreover, note that there also exists a dummy fitness assignment strategy in case it would be usefull for some specific implementation.

Achievement Fitness Assignment Scheme. One of the provided fitness assignment schemes is the familiy of *achievement (scalarizing) functions*, initially proposed by Wierzbicki in [23]. This a posteriori scalar approach, used by many MCDM methods, is based on an arbitrary reference point R (generally given by the DM) and consists in projecting R onto the set of Pareto optimal solutions. The problem to be solved is then converting into a mono-objective one.

Fast Non-dominated Sorting Fitness Assignment Scheme. Another implemented fitness assignment scheme is the dominance depth proposed in [21] and used, for instance, in NSGA (Non-dominated Sorting Genetic Alorithm) [21] and NSGA-II [5]. This strategy, based on the Goldberg's version of Pareto-ranking [9], consits in classifying a set of solutions onto several classes (or fronts). A solution that belongs to a class does not dominate another

one from the same class. Then, individuals from the first front all belong to the best non-dominated set of the population; individuals from the second front all belong to the second best non-dominated set; and so on. In a logical way, the best fitness value is assigned to solutions of the first class, because they are closest to the true Pareto-optimal front of the problem. This scheme naturally tends to search for solutions located in non-dominated regions.

Indicator-Based Fitness Assignment Scheme. Introduced by Zitzler and Künzli [24] and used, for instance, in IBEA (Indicator-Based Evolutionary Algorithm), this startegy has the characteristic to compute fitness values by comparing individuals on the basis of an arbitrary binary quality indicator I (also called binary performance metric). Thereby, no particular diversity preservation mechanism is necessary depending on the specified indicator. This one, that can be determined according to the DM preferences, denotes the overall goal of the optimization process. Thus, the fitness of a solution measures its usefulness according to this optimization goal. As discussed later in the paper, several binary quality indicators to be used with this fitness assignment scheme are proposed within ParadisEO-MOEO.

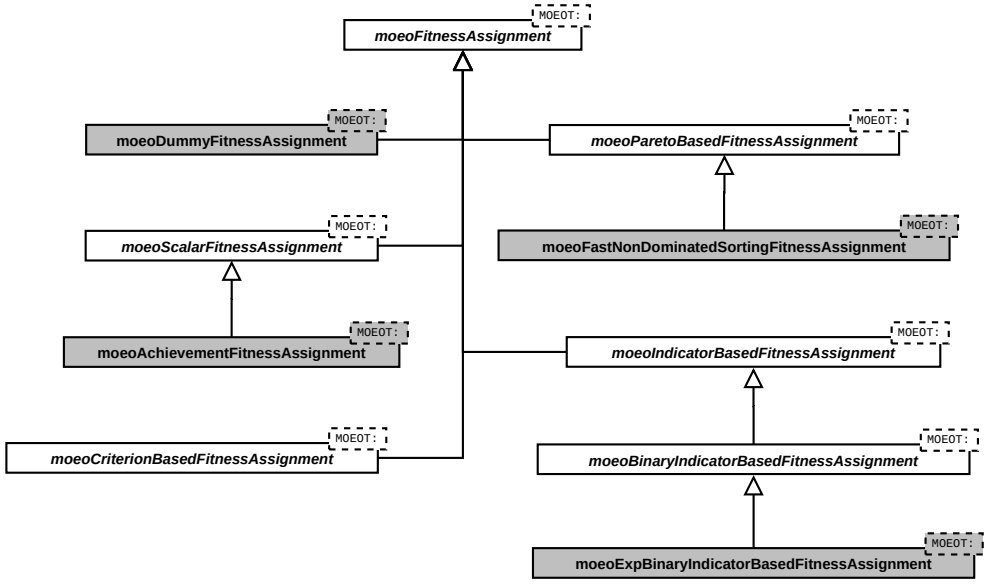

Figure 3: Core classes for fitness assignment schemes.

3.1.3 Diversity Assignment Schemes

Aiming at approximating the Pareto optimal set is not only a question of convergence. The final approximation have to be both well-converged and well-diversified. However, a classical Pareto-based fitness assignment scheme often tends to produce premature convergence by favoring non-dominated solutions, what does not guarantee a uniformly sampled output set. To avoid that, a diversity maintaining mechanism is usually employed to distribute the population over the Pareto-optimal region.

Sharing Diversity Assignment Scheme. *Sharing* (or *fitness sharing*), initially suggested by Goldberg and Richardson [10] to preserve diversity amoung solutions in an EA population, has first been employed by Fonseca and Fleming [7] in the case of MOO. This *Kernel* method consists in estimating the distribution density of a solution using a so-called *sharing function* that is related to the sum of distances to its neighborhood solutions. A sharing distance parameter specifies the similarity thereshold (*i.e.* the size of *niches*). The distance measure between two solutions can be definied either in the decision space, in the objective space or can combined both. A distance metric partly or fully definied in the parameter space strongly depends of the tackled problem. But, standard normalized distance metrics defined in the criterion space are already provided within ParadisEO-MOEO for real-coded objective vectors. Sharing is one of the most popular technique and is commonly used in a large number of MOMHs such as MOGA [7], NPGA [12], NSGA [21] and more. Note that, there exists a 'front by front' procedure as, in some cases (*e.g.* in MOGA [7] and NSGA [21]), sharing only occurs between solutions of same rank.

Crowding Diversity Assignment Scheme. Another diversity assignment scheme is the concept of *Crowding*, firstly suggested by Holland [11] and used by De Jong [13] to prevent *genetic drift* [13]. It has been succefully applied by Deb et al. [5] in the case of MOO in their NSGA-II (Non-dominated Sorting Genetic Alorithm II). Contrary to the sharing diversity assignment scheme, this one allows to maintain diversity without specifying any parameters. It consits in estimating the density of solutions surrounding a particular point of the objective space. As previously, a similar mechanism working on sub-classes of solutions is also provided within ParadisEO-MOEO.

3.1.4 Elitism

Another essential point about Pareto-based resolution is the concept of *elitism*. This concept is very popular since the end of the nineties and is used in a big majority of recent MOMHs (*e.g.* [17, 25]). It consists in maintaining an external set, the so-called *archive*, that allows to store all the non-dominated or the most prefered solutions found during the search. This archive mainly aims at preventing that these solutions are not lost during the (often stochastic) optimization process. As shown in figure 5, in term of implementation, an archive is an unbounded population using a particular dominance relation to update its contents. The Pareto dominance criterion is the most commonly employed and is used as default, but

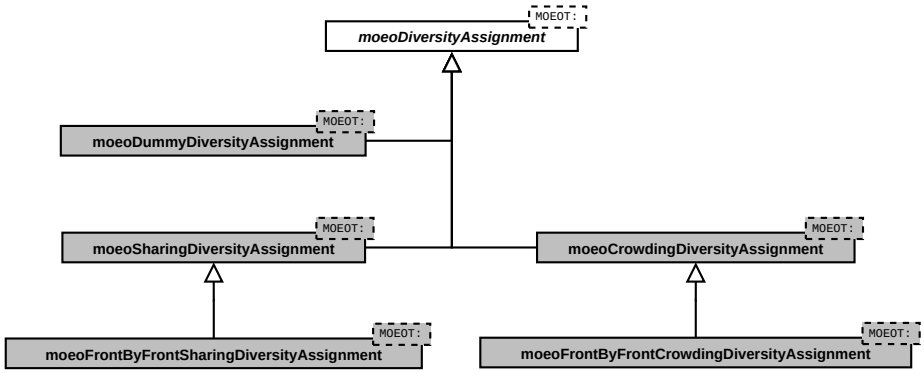

Figure 4: Core classes for diversity assignment schemes.

ParadisEO-MOEO also gives the oportunity to use any other dominance relation (as shown in figure 6). The archive is basically used as an external storage. As well, we will see in the next subsection that its members can also be included during the selection phase of a MOEA.

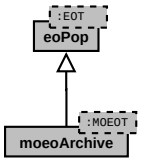

Figure 5: Core classes for archiving.

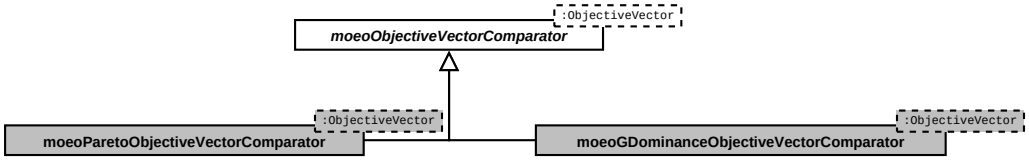

Figure 6: Core classes for dominance relation (used for a parwise objective vector comparison).

3.1.5 Statistical Tools

Some usefull statistical tools are also provided within ParadisEO-MOEO. Then, it is for instance possible to save the contents of the archive at each iteration, so that the evolution

of the non-dominated set can be observed or study using graphical tools such as GUIMOO[2]. Furthermore, as pointed out in section 2, an important issue in MOO is the performance analysis, commonly done by using performance metrics. As shown in figure 7, a couple of metrics are featured within ParadisEO-MOEO. Some are used to quantify the quality of a Pareto set (or of a solution), while other are used for pairwise comparisons (between two Pareto sets or two solutions). The entropy [1] and the contribution [18] metrics are both already implemented and can thus be used to compare two sets of solutions. Moreover, it is possible to compare the current archive with the archive of the previous iteration using a binary metric, and to print the progression of this measure iteration after iteration. Besides, some metrics for pairwise comparison of solutions (that are then useable with the binary indicator-based fitness assignment schemes) are proposed: the additive ϵ-indicator [27] and the I_{HD}-indicator [27] (based on the hypervolume concept introduced in [26]). Of course, other metrics can easily be implemented and a few will soon be.

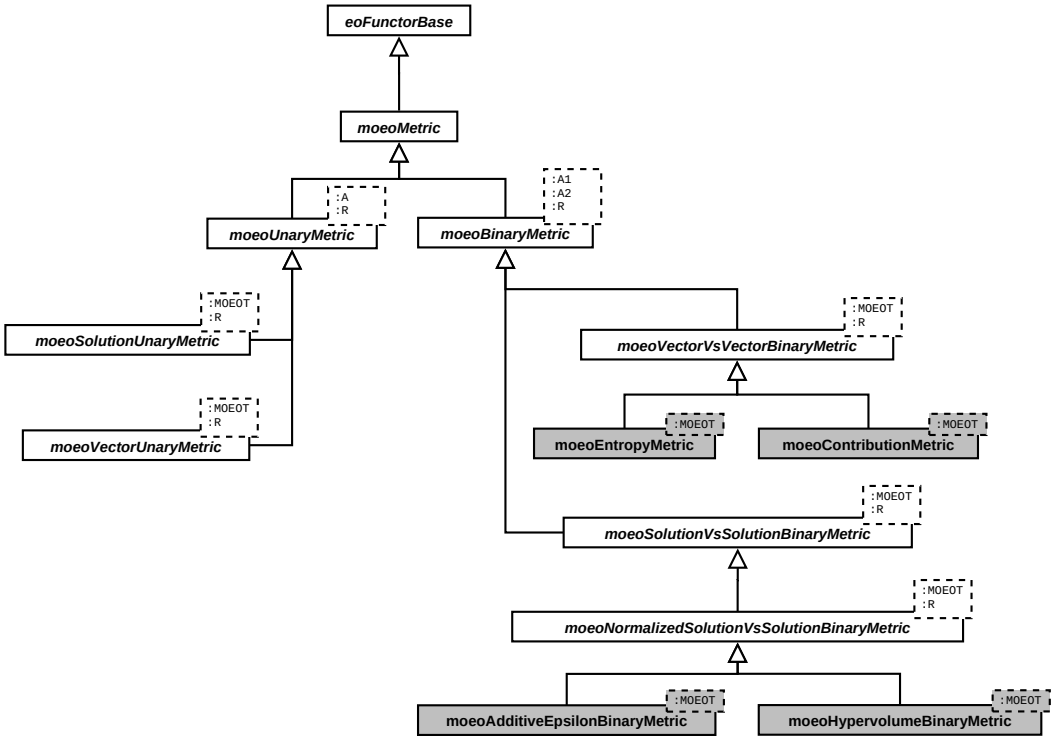

Figure 7: Core classes for metrics.

[2]GUIMOO is a Graphical User Interface for Multi-Objective Optimization available at `http://guimoo.gforge.inria.fr/`

3.2 MOEA-related Components

As a MOEA only differs of a mono-objective EA in a fiew points, many of the EO [16] components are reusable within ParadisEO-MOEO. The reader is refered to [16] for more details about the MOO-independent components of ParadisEO. Using EO [16] and ParadisEO-MOEO, it is possible to build a complete and efficient multi-objective optimization solver. Generally speaking, the two main differences between a MOEA and a mono-objective EA appears during the selection and the replacement steps of the algorithms. Then, as the representation in the objective space do not correspond to a scalar value in the multi-objective case, EO's selection and replacement schemes cannot be used in a MOO application. Therefore, the major contributions of this framework refer to i) fitness assignment, diversity preservation and elitism and to ii) the selection and replacement schemes that need to be defined for the specific case of MOO, where the fitness and diversity values are used instead of a single scalar objective value (as it is generally the case in mono-objective optimization). Of course, the representation of a solution as well as the way to build an EA also needs to be redefined for the multi-objective case.

3.2.1 Selection Schemes

As noticed earlier, two of the main differences between a mono-objective EA and a MOEA appears on the selection and the replacement steps of the algorithm, during which are usually used the fitness and/or the diversity value(s). The selection step consists in choosing some solutions that will be used to generate the offspring population. There exists a large number of selection strategies in the case of MOO, four ones are already provided within ParadisEO-MOEO (see figure 8):

A *random selection scheme*, that consists in selecting a parent randomly among the population's members, without taking the fitness values nor the diversity values into account.

A *deterministic tournament selection scheme*, that consists in performing a tournament between n randomly chosen population's members and in selecting the best one.

A *stochastic tournament selection scheme*, that consists in performing a binary tournament between randomly chosen population's members and in selecting the best one with a probability p or the worst one with a probability $(1 - p)$.

A *selection scheme from the population and the archive*, that consists in selecting a population's member using a particular selection scheme with a probability p or in selecting an archive's member using another particular selection scheme with a probability $(1 - p)$. So, non-dominated or most-prefered solutions also contributes to the evolution engine by being used to create the offspring. This scheme has successfully been applied in various elitist MOEAs with efficient performances (experimentally speaking), as it the case, for instance, in SPEA [26] or SPEA2 [25] among others.

Of course, everything is done to easily implement a new selection scheme with a minimum programming effort.

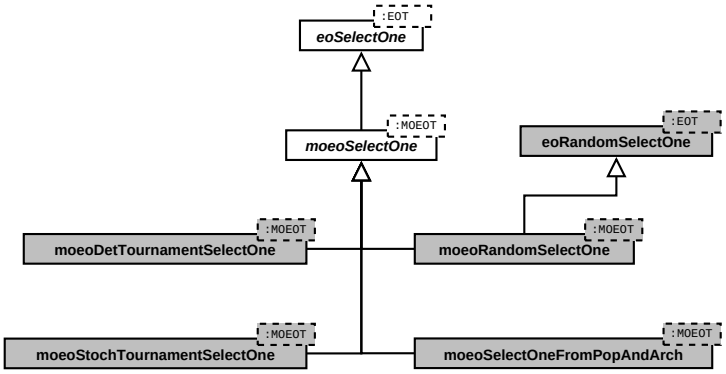

Figure 8: Core classes for selection schemes.

3.2.2 Replacement Schemes

The replacement step of an EA consists in choosing which solutions will be part of the next generation by using both the offspring and the parent populations. The majority of the replacement strategies depends on the fitness and/or the diversity value(s) and are then MOO-specific. Three replacement schemes are already provided within ParadisEO-MOEO (see figure 9), but this is not exhaustive as new ones can easilly be implemented due to the genericity of the framework:

A *generational replacement scheme*, that consists in keeping the offspring population only, while all parents are deleted.

An *elitist replacement scheme*, that consists in choosing the N best solutions (where N is the population size).

An *environmental replacement scheme*, that consists in deleting one-by-one the worst individuals, and in updating the fitness and the diversity values of the remaining solutions each time their is a deletion. The processus ends when the desired population size is reached.

3.2.3 Multi-Objective Evolutionary Algorithms

Now that all the MOO-specific components are definied, a MOEA can easily be developped using the fine-grained classes of both EO [16] and ParadisEO-MOEO. As the implementation is conceptually divided into components, different operators can be experimented without

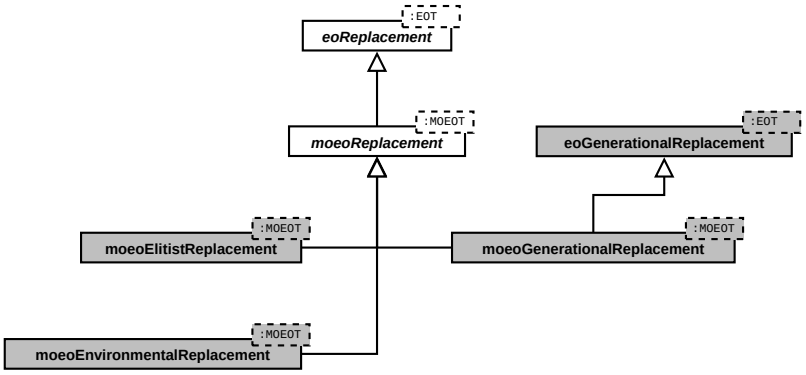

Figure 9: Core classes for replacement schemes.

engendering significant modifications in term of code writing. As seen before, a wide range of components are already provided. But, keep in mind that this list is not exhaustive as the framework perpetually evolves and offers all that is necessary to develop new ones with a minimum effort, as ParadisEO is a white-box framework that tends to be flexible while being as user-friendly as possible.

Figure 10 illustrates the use of the `moeoEasyEA` class that allows to define a MOEA in a common fashion by specifying all the particular components required for its implementation. As often within ParadisEO, all the classes use a template parameter *MOEOT (Multi-Objective Evolving Object Type)* that defines the type of a solution for the problem under consideration. This type might be represented by inheriting of the `MOEO` class that is described in subsection 3.1.1.

In order to satisfy both the common user and the more experimented one, ParadisEO-MOEO also provides even more easy-to-use MOEAs (see figure 11). These classes propose an implementation of standard MOEAs such as NSGA [21], NSGA-II [5] or IBEA [24] by using the fine-grained components of ParadisEO while requiring a minimum number of problem- or algorithm-specific parameters. These easy-to-use MOEAs also tends to be used as state-of-the-art references for a fair performance comparison in the academic world, even if they are also well-suited for a straight use on a real-world problem. In a close future, other easy-to-use MOEAs will be proposed while new required components will be implemented within ParadisEO-MOEO.

4 Applications

ParadisEO-MOEO has been experimented on different both academic and industrial problems. In this section, we present three applications that show the wide range of potential of this framework as it has succesfully been applied to scheduling problems, continuous optimization and data-mining applications.

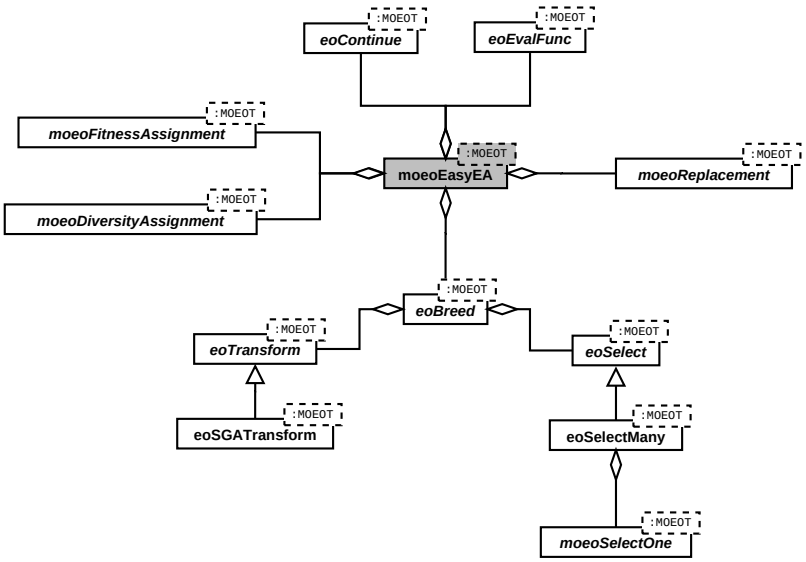

Figure 10: Core classes required by `moeoEasyEA` to design a MOEA

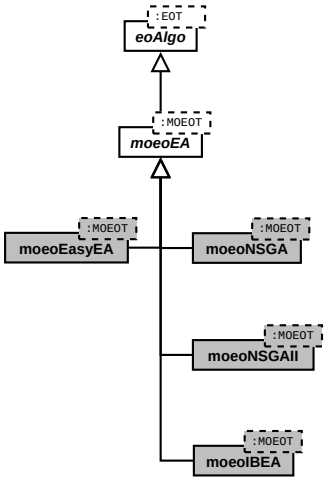

Figure 11: Core classes for easy-to-use evolutionary algorithms.

A bi-objective flow-shop scheduling problem. The flow-shop is one of the most widely investigated scheduling problem of the literature. But, the majority of studies considers it on a single-criterion form. However, other objectives than minimizing the makespan can be taken into account, like, *e.g.*, minimizing the total tardiness.

Electromagnetic properties of conducting polymer composites in the microwave band. Due to the proliferation of electromagnetic interferences, designing protecting material for high frequencies equipments has become an important problem. In [15], a new multi-objective model is proposed to design the different layers of a conducting polymer. To solve this model, a multi-objective continuous genetic algorithm is used. This algorithm offers several solutions with different physical properties and different costs.

Knowledge discovery in biological data from microarray experiments. The problem of analyzing microarray data is actually a major issue in genomics. Often used techniques are clustering and classification. In [14], the authors propose to analyze those data through association rules. The problem is modeled as a multi-objective rule mining problem and a genetic algorithm is used to explore the large search space associated.

5 Conclusion and Perspectives

In this paper, we introduced ParadisEO-MOEO, a framework dedicated to the reusable design of evolutionary multi-objective optimization applications[3]. ParadisEO-MOEO is based on the EO library [16]. It provides the most common multi-objective fitness assignment shemes and diversity preservation mechanisms, as well as a wide range of archive-related features such as non-dominated solutions storage, elitism and performance metrics computation. This framework has been used to solve many type of applications, from academic to real-world problems.

ParadisEO-MOEO is an open-source white-box object-oriented framework that aims to simplify and speed up the incremental implementation of a whole efficient multi-objective optimization program. In order to confer a maximum design and code reuse, it is based on a clear conceptual distinction between the metaheuristics and the problem representations. This separation is expressed at the implementation level, and the hierarchical classes that are provided allow the designer to extend the framework by inheritance or specialization. Furthermore, the fine-grained components of ParadisEO-MOEO confer a high flexibility compared to other frameworks. Modifying existing components or adding new ones can easily be done without impacting the whole application. Besides, ParadisEO-MOEO is a part of the ParadisEO framework that covers the most common parallel/distributed models and hybridization mechanisms. The user can thus directly include some ParadisEO features into an application designed using ParadisEO-MOEO in a fast and simple way.

[3] ParadisEO-MOEO is available at `http://paradiseo.gforge.inria.fr`

In the future, ParadisEO-MOEO needs to constantly evolve in order to reflect the advances of the literature. New fitness assignment strategies as well as new performance metrics for Pareto set approximations should also be proposed before long. Moreover, a major extension of ParadisEO-MOEO would be to allow the design of exact methods as well as their hybridization with already provided metaheuristics. Besides, it would be interesting to introduce new specific concepts emerging from multi-criterion optimization such as the consideration of uncertainty through stochastic or fuzzy multi-objective problems.

References

[1] M. Basseur, F. Seynhaeve, and E.-G. Talbi. Design of multi-objective evolutionary algorithms: Application to the flow-shop scheduling problem. In *Congress on Evolutionary Computation (CEC'02)*, volume 2, pages 1151–1156, Honolulu, Hawaii, USA, 2002.

[2] S. Bleuler, M. Laumanns, L. Thiele, and E. Zitzler. PISA — a platform and programming language independent interface for search algorithms. In C. M. Fonseca, P. J. Fleming, E. Zitzler, K. Deb, and L. Thiele, editors, *Evolutionary Multi-Criterion Optimization, Second International Conference (EMO'03)*, volume 2632 of *Lecture Notes in Computer Science*, pages 494–508, Faro, Portugal, 2003. Springer-Verlag.

[3] C. A. Coello Coello, D. A. Van Veldhuizen, and G. B. Lamont. *Evolutionary Algorithms for Solving Multi-Objective Optimization Problems*. Kluwer Academic Publishers, 2002.

[4] K. Deb. *Multi-Objective Optimization using Evolutionary Algorithms*. Wiley, 2001.

[5] K. Deb, S. Agrawal, A. Pratab, and T. Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In *Proceedings of the 6th International Conference on Parallel Problem Solving from Nature (PPSN VI)*, pages 849–858, Paris, France, 2000.

[6] M. Emmerich and R. Hosenberg. TEA - a toolbox for the design of parallel evolutionary algorithms in c++. Technical Report CI-106/01, SFB 531, University of Dortmund, Germany, 2001.

[7] C. M. Fonseca and P. J. Fleming. Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. In S. Forrest, editor, *Proceedings of the 5th International Conference on Genetic Algorithms*, pages 416–423, San Mateo, California, USA, 1993. Morgan Kaufmann publishers.

[8] C. Gagné and M. Parizeau. Genericity in evolutionary computation software tools: Principles and case study. *International Journal on Artificial Intelligence Tools*, 15(2):173–194, 2006.

[9] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, 1989.

[10] D. E. Goldberg and J. Richardson. Genetic algorithms with sharing for multimodal function optimization. In *Proceedings of the Second International Conference on Genetic Algorithms and their application*, pages 41–49, Mahwah, NJ, USA, 1987. Lawrence Erlbaum Associates, Inc.

[11] J. H. Holland. *Adaptation in Natural and Artificial Systems.* University of Michigan Press, Ann Abor, MI, USA, 1975.

[12] J. Horn, N. Nafpliotis, and D. E. Goldberg. A niched pareto genetic algorithm for multiobjective optimization. In *In Proceeding of The First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence*, volume 1, pages 82–87, 1994.

[13] K. A. De Jong. *An Analysis of the Behavior of a Class of Genetic Adaptive Systems.* PhD thesis, Ann Arbor, University of Michigan, 1975.

[14] L. Jourdan, M. Khabzaoui, C. Dhaenens, and E.-G. Talbi. A hybrid evolutionary algorithm for knowledge discovery in microarray experiments. In S. Olariu and A. Y. Zomaya, editors, *Handbook of Bioinspired Algorithms and Applications*, chapter 28, pages 489–505. CRC Press, 2005.

[15] L. Jourdan, T. Legrand, E.-G. Talbi, and J.-L. Wojkiewicz. Mono and multi-objective continuous optimization for conducting polymer composites. In *ECCO XIX / CO 2006*, Porto, Portugal, 2006.

[16] M. Keijzer, J.-J. Merelo, G. Romero, and M. Schoenauer. Evolving objects: A general purpose evolutionary computation library. In *Proceedings of the 5th International Conference on Artificial Evolution (EA'01)*, pages 231–244, Le Creusot, France, 2001. http://eodev.sourceforge.net.

[17] J. D. Knowles and D. Corne. Approximating the nondominated front using the pareto archived evolution strategy. *Evolutionary Computation*, 8(2):149–172, 2000.

[18] H. Meunier, E.-G. Talbi, and P. Reininger. A multiobjective genetic algorithm for radio network optimization. In *Proceedings of the 2000 Congress on Evolutionary Computation (CEC'00)*, pages 317–324. IEEE Press, 2000.

[19] K. Miettinen. *Nonlinear Multiobjective Optimization*, volume 12. Kluwer Academic Publishers, Boston, USA, 1999.

[20] V. Pareto. *Cours d'Economie Poliitique.* Rouge, Lausanne, Switzerland, 1896.

[21] N. Srinivas and K. Deb. Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, 2(3):221–248, 1994.

[22] K. C. Tan, T. H. Lee, D. Khoo, and E. F. Khor. A multi-objective evolutionary algorithm toolbox for computer-aided multi-objective optimization. *IEEE Transactions on Systems, Man and Cybernetics: Part B (Cybernetics)*, 31(4):537–556, 2001.

[23] A. Wierzbicki. The use of reference objectives in multiobjective optimization. In G. Fandel and T. Gal, editors, *Multiple Objective Decision Making, Theory and Application*, number 177 in Lecture notes in economics and mathematical systems, pages 468–486. Springer-Verlag, 1980.

[24] E. Zitzler and S. Künzli. Indicator-based selection in multiobjective search. In *Proceedings of the 8th International Conference on Parallel Problem Solving from Nature (PPSN VIII)*, volume 3242 of *Lecture Notes in Computer Science*, pages 832–842, Birmingham, UK, 2004. Springer-Verlag.

[25] E. Zitzler, M. Laumanns, and L.Thiele. SPEA2: Improving the strength pareto evolutionary algorithm. Technical Report 103, Computer Engineering and Networks Lab (TIK), Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, 2001.

[26] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, 1999.

[27] E. Zitzler, L. Thiele, M. Laumanns, C. M. Foneseca, and V. Grunert da Fonseca. Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Transactions on Evolutionary Computation*, 7(2):117–132, 2003.

Unité de recherche INRIA Futurs
Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)

Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)

ISSN 0249-6399