A Unified Model for Evolutionary Multiobjective Optimization and its Implementation in a General Purpose Software Framework: ParadisEO-MOEO

Arnaud Liefooghe, Laetitia Jourdan, El-Ghazali Talbi

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

A Unified Model for Evolutionary Multiobjective Optimization and its Implementation in a General Purpose Software Framework: ParadisEO-MOEO

Arnaud Liefooghe — Laetitia Jourdan — El-Ghazali Talbi

N° 6906

April 2009

Thème NUM

ISSN 0249-6399 ISRN INRIA/RR--6906--FR+ENG

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

INRIA

centre de recherche
LILLE - NORD EUROPE

A Unified Model for Evolutionary Multiobjective Optimization and its Implementation in a General Purpose Software Framework: ParadisEO-MOEO

Arnaud Liefooghe , Laetitia Jourdan , El-Ghazali Talbi

Thème NUM — Systèmes numériques
Équipe-Projet Dolphin

Rapport de recherche n° 6906 — April 2009 — 28 pages

Abstract: This paper gives a concise overview of evolutionary algorithms for multiobjective optimization. A substantial number of evolutionary computation methods for multiobjective problem solving has been proposed so far, and an attempt of unifying existing approaches is here presented. Based on a fine-grained decomposition and following the main issues of fitness assignment, diversity preservation and elitism, a conceptual global model is proposed and is validated by regarding a number of state-of-the-art algorithms as simple variants of the same structure. The presented model is then incorporated into a general-purpose software framework dedicated to the design and the implementation of evolutionary multiobjective optimization techniques: ParadisEO-MOEO. This package has proven its validity and flexibility by enabling the resolution of many real-world and hard multiobjective optimization problems.

Key-words: evolutionary algorithms, multiobjective optimization, conceptual unified model, algorithm design and implementation, software frameworks

Centre de recherche INRIA Lille – Nord Europe
Parc Scientifique de la Haute Borne
40, avenue Halley, 59650 Villeneuve d'Ascq
Téléphone : +33 3 59 57 78 00 — Télécopie : +33 3 59 57 78 50

Un model unifié pour l'optimisation évolutionnaire multiobjectif et son implémentation dans un cadre logiciel générique: ParadisEO-MOEO

Résumé : Ce document donne un aperu concis des algorithmes évolutionnaires pour l'optimisation multiobjectif. Un nombre conséquent de méthodes évolutionnaires dédiées à la résolution de problèmes multiobjectifs a été proposé à ce jour, et une tentative d'unifier les approches existantes est ici présentée. Sur la base d'une décomposition à grain fin et suite à la description des principales questions de *fitness assignment*, de préservation de la diversité et d'élitisme, un modèle conceptuel est proposé et validé en traitant un certain nombre algorithmes classiques comme de simples variantes de la même structure. Le modèle présenté est alors incorporé dans un logiciel dédié à la conception et à l'implémentation d'algorithmes évolutionnaires pour l'optimisation multiobjectif: Paradiseo-MOEO. Cette librairie a démontré sa validité et sa grande souplesse en permettant la résolution d'un grand nombre de problèmes d'optimisation multiobjectifs réels et difficiles.

Mots-clés : algorithmes évolutionnaires, optimisation multiobjectif, modèle conceptuel unifié, conception et implémentation d'algorithme, cadre logiciel

1 Introduction

Evolutionary Multiobjective Optimization (EMO) is one of the most challenging areas in the field of multicriteria decision making. Generally speaking, a Multiobjective Optimization Problem (MOP) can be defined by a vector function f of $n \geq 2$ objective functions $(f_1, f_2, \ldots, f_n)$, a set X of feasible solutions in the *decision space*, and a set Z of feasible points in the *objective space*. Without loss of generality, we assume that $Z \subseteq \mathbb{R}^n$ and that all n objective functions are to be minimized. To each decision vector $x \in X$ is assigned an objective vector $z \in Z$ on the basis of the vector function $f : X \rightarrow Z$ with $z = f(x)$. A dominance relation is then usually assumed so that a partial order is induced over X. Numerous dominance relations exist in the literature and will be discussed later in the paper. Let us consider the well-known concept of *Pareto dominance*, for which a given objective vector $z \in Z$ is said to *dominate* another objective vector $z' \in Z$ if $\forall i \in \{1, 2, \ldots, n\}$, $z_i \leq z'_i$ and $\exists j \in \{1, 2, \ldots, n\}$ such as $z_j < z'_j$. An objective vector $z \in Z$ is said to be *nondominated* if there does not exist any other objective vector $z' \in Z$ such that z' dominates z. By extension, we will say that a decision vector $x \in X$ *dominates* a decision vector $x' \in X$ if $f(x)$ dominates $f(x')$, and that a decision vector $x \in X$ is *nondominated* (or *efficient*, *Pareto optimal*) if $f(x)$ maps to a nondominated point. The set of all efficient solutions is called *efficient* (or *Pareto optimal*) *set* and its mapping in the objective space is called *Pareto front*.

In practice, different resolution scenarios exist and strongly rely on the cooperation between the search process and the decision making process. Indeed, a distinction can be made between the following forms such a cooperation might take. For instance, the Decision Maker (DM) may be interested in identifying the whole set of efficient solutions, in which case the choice of the most preferred solution is made *a posteriori*. However, when preference information can be provided *a priori*, the search may lead to the potential best compromise solution(s) over a particular preferred region of the Pareto front. A third class of methods consists of a progressive, interactive, cooperation between the DM and the solver. However, in any case, the overall goal is often to identify a set of good-quality solutions. But generating such a set is usually infeasible, due to the complexity of the underlying problem or to the large number of optima. Therefore, the overall goal is often to identify a good approximation of it. Evolutionary algorithms are commonly used to this end, as they are particularly well-suited to find multiple efficient solutions in a single simulation run. The reader is referred to [1, 2] for more details about EMO.

As pointed out by different authors (see *e.g.* [2, 3]), approximating an efficient set is itself a bi-objective problem. Indeed, the approximation to be found must have both good convergence and distribution properties, as its mapping in the objective space has to be (*i*) close to, and (*ii*) well-spread over the (generally unknown) optimal Pareto front, or a subpart of it. As a consequence, the main difference between the design of a single-objective and of a multiobjective search method deals with these two goals. Over the last two decades, major advances, from both algorithmic and theoretical aspect, have been made in the EMO field. And a large number of algorithms have been proposed. Among existing approaches, one may cite VEGA [4], MOGA [5], NSGA [6], NSGA-II [7], NPGA [8], SPEA [9], SPEA2 [10] or PESA [11]. All these methods are presented and described in [2]. Note that another topic to mention while dealing

with EMO relates to performance assessment. Various quality indicators have been proposed in the literature for evaluating the performance of multiobjective search methods. The reader is referred to [12] for a review.

In [3], Zitzler et al. notice that initial EMO approaches were mainly focused on moving toward the Pareto front [4, 13]. Afterwards, diversity preservation mechanisms quickly emerged [5, 6, 8]. Then, at the end of the nineteens, the concept of elitism, related to the preservation of nondominated solutions, became very popular and is now employed in most recent EMO methods [9, 10, 14]. Specific issues of *fitness assignment*, *diversity preservation* and *elitism* are commonly approved in the community and are also presented under different names in, for instance, [2, 3]. Based on these three main notions, several attempts have been made in the past for unifying EMO algorithms. In [15], the authors focus on elitist EMO search methods. This study has been later extended in [3] where the algorithmic concepts of fitness assignment, diversity preservation and elitism are largely discussed. More recently, Deb proposed a robust framework for EMO [16] based on NSGA-II (Non-dominated Sorting Genetic Algorithm) [7]. The latter approach is decomposed into three main EMO-components related to elite preservation, nondominated solutions emphasis and diversity maintaining. However, this model is strictly focused on NSGA-II, whereas other state-of-the-art methods can be decomposed in the same way. Indeed, a lot of components are shared by many EMO algorithms, so that, in somehow, they can all be seen as variants of the same unified model, as it will be highlighted in the remainder of the paper. Furthermore, some existing models have been used as a basis for the design of tools to help practitioners for MOP solving. For instance, following [3, 15], the authors proposed a software framework for EMO called PISA [17]. PISA is a platform and programming language independent interface for search algorithms that consists of two independent modules (the variator and the selector) communicating via text files. Note that other software frameworks dealing with the design of metaheuristics for EMO have been proposed, including jMetal [18], the MOEA toolbox for Matlbax [19], MOMHLib++ [20] and Shark [21]. These packages will be discussed later in the paper.

The purpose of the present work is twofold. Firstly, a unified view of EMO is given. We describe the basic components shared by many algorithms, and we introduce a general purpose model as well as a classification of its fine-grained components. Next, we confirm its high genericity and modularity by treating a number of state-of-the-art methods as simple instances of the model. NSGA-II [7], SPEA2 [10] and IBEA [22] are taken as examples. Afterwards, we illustrate how this general-purpose model has been used as a starting point for the design and the implementation of an open-source software framework dedicated to the reusable design of EMO algorithms, namely ParadisEO-MOEO[1]. All the implementation choices have been strongly motivated by the unified view presented in the paper. This free C++ white-box framework has been widely experimented and has enabled the resolution of a large diversity of MOPs from both academic and real-world applications. In comparison to the literature, we expect the proposed unified model to be more complete, to provide a more fine-grained decomposition, and the software framework to offer a more modular implementation than previous similar attempts. The reminder of the paper is organized as follows. In Sect. 2, a concise, unified and up-to-date presentation

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

of EMO techniques is discussed. Next, a motivated presentation of the software framework introduced in this paper is given in Sect. 3, and is followed by a detailed description of the design and the implementation of EMO algorithms under ParadisEO-MOEO. Finally, the last section concludes the paper.

2 The Proposed Unified Model

An Evolutionary Algorithm (EA) [23] is a search method that belongs to the class of metaheuristics [24], and where a population of solutions is iteratively improved by means of some stochastic operators. Starting from an initial population, each individual is evaluated in the objective space and a selection scheme is performed to build a so-called parent population. An offspring population is then created by applying variation operators. Next, a replacement strategy determines which individuals will survive. The search process is iterated until a given stopping criterion is satisfied.

As noticed earlier in the paper, in the frame of EMO, the main expansions deal with the issues of *fitness assignment*, *diversity preservation* and *elitism*. Indeed, contrary to single-objective optimization where the fitness value of a solution corresponds to its single objective value in most cases, a multiobjective fitness assignment scheme is here required to assess the individuals performance, as the mapping of a solution in the objective space is now multi-dimensional. Moreover, trying to approximate the efficient set is not only a question of convergence. The final approximation also has to be well spread over the objective space, so that a diversity preservation mechanism is usually required. This fitness and diversity information is necessary to discriminate individuals at the selection and the replacement steps of the EA. Next, the main purpose of elitism is to avoid the loss of best-found nondominated solutions during the stochastic search process. These solutions are frequently incorporated into a secondary population, the so-called *archive*. The update of the archive contents possibly appear at each EA iteration.

As a consequence, whatever the MOP to be solved, the common concepts for the design of an EMO algorithm are the following ones:

1. Design a representation.
2. Design a population initialization strategy.
3. Design a way of evaluating a solution.
4. Design suitable variation operators.
5. Decide a fitness assignment strategy.
6. Decide a diversity preservation strategy.
7. Decide a selection strategy.
8. Decide a replacement strategy.
9. Decide an archive management strategy.
10. Decide a continuation strategy.

When dealing with any kind of metaheuristics, one may distinguish problem-specific and generic components. Indeed, the former four common-concepts presented above strongly depends of the MOP at hand, while the six latter ones can be considered as problem-independent, even if some problem-dependent strategies can also be envisaged in some particular cases. Note that concepts of representation and evaluation are shared by any metaheuristic, concepts of population initialization and stopping criterion are shared by any population-based metaheuristic, concepts of variation operators, selection and replacement are shared by any EA, whereas concepts of fitness, diversity and archiving are specific to EMO.

2.1 Components Description

This section provides a description of components involved in the proposed unified model. EMO-related components are detailed in more depth.

2.1.1 Representation

Solution representation is the starting point for anyone who plans to design any kind of metaheuristic. A MOP solution needs to be represented both in the decision space and in the objective space. While the representation in the objective space can be seen as problem-independent, the representation in the decision space must be relevant to the tackled problem. Successful applications of metaheuristics strongly requires a proper solution representation. Various encodings may be used such as binary variables, real-coded vectors, permutations, discrete vectors, and more complex representations. Note that the choice of a representation will considerably influence the way solutions will be initialized and evaluated in the objective space, and the way variation operators will be applied.

2.1.2 Initialization

Whatever the algorithmic solution to be designed, a way to initialize a solution (or a population of solutions) is expected. While dealing with any population-based metaheuristic, one has to keep in mind that the initial population must be well diversified in order to prevent a premature convergence. This remark is even more true for MOPs where the goal is to find a well-converged and a well-spread approximation. The way to initialize a solution is closely related to the problem under consideration and to the representation at hand. In most approaches, the initial population is generated randomly or according to a given diversity function.

2.1.3 Evaluation

The problem at hand is to optimize a set of objective functions simultaneously over a given search space. Then, each time a new solution integrates the population, its objective vector must be evaluated, *i.e.* the value corresponding to each objective function must be set.

2.1.4 Variation

The purpose of variation operators is to modify the representation of solutions in order to move them in the search space. Generally speaking, while dealing with EAs, these problem-dependent operators are stochastic. Mutation operators are unary operators acting on a single solution whereas recombination (or crossover) operators are mostly binary, and sometimes n-ary.

2.1.5 Fitness Assignment

In the single-objective case, the fitness value assigned to a given solution is most often its unidimensional objective value. While dealing with MOPs, fitness assignment aims to guide the search toward Pareto optimal solutions for a better convergence. Extending [2, 3], we propose to classify existing fitness assignment schemes into four different families:

- *Scalar approaches*, where the MOP is reduced to a single-objective optimization problem. A popular example consists of combining the n objective functions into a single one by means of a weighted-sum aggregation. Other examples are ϵ-constraint or achievement function-based methods [25].

- *Criterion-based approaches*, where each objective function is treated separately. For instance, in VEGA (Vector Evaluated GA) [4], a parallel selection is performed where solutions are discerned according to their values on a single objective function, independently to the others. In lexicographic methods [13], a hierarchical order is defined between objective functions.

- *Dominance-based approaches*, where a dominance relation is used to classify solutions. For instance, *dominance-rank* techniques computes the number of population items that dominate a given solution [5]. Such a strategy take part in, *e.g.*, Fonseca and Fleming MOGA (Multiobjective GA) [5]. In *dominance-count* techniques, the fitness value of a solution corresponds to the number of individuals that are dominated by these solutions [9]. Finally, *dominance-depth* strategies consists of classifying a set of solutions into different classes (or fronts) [26]. Hence, a solution that belongs to a class does not dominate another one from the same class; so that individuals from the first front all belong to the best nondominated set, individuals from the second front all belong to the second best nondominated set, and so on. The latter approach is used in NSGA (Nondominated Sorting GA) [6] and NSGA-II [7]. However, note that several schemes can also be combined, what is the case, for example, in [9]. In the frame of dominance-based approaches, the most commonly used dominance relation is based on Pareto-dominance as given in Sect. 1. But some recent techniques are based on other dominance operators such as ϵ-dominance in [27] or g-dominance in [28].

- *Indicator-based approaches*, where the fitness values are computed by comparing individuals on the basis of a quality indicator I. The chosen indicator represents the overall goal of the search process. Generally speaking, no particular diversity preservation mechanism is in usual necessary, with

regards to the indicator being used. Examples of indicator-based EAs are IBEA (Indicator-Based EA) [22] or SMS-EMOA (S-Metric Selection EMO Algorithm) [29].

2.1.6 Diversity Assignment

As noticed in the previous section, aiming at approximating the efficient set is not only a question of convergence. The final approximation also has to be well spread over the objective space. However, classical dominance-based fitness assignment schemes often tend to produce premature convergence by privileging nondominated solutions, what does not guarantee a uniformly sampled output set. In order to prevent that issue, a diversity preservation mechanism, based on a given distance measure, is usually integrated into the algorithm to uniformly distribute the population over the trade-off surface. In the frame of EMO, a common distance measure is based on the euclidean distance between objective vectors. But, this measure can also be defined in the decision space or can even combined both spaces. Popular examples of EMO diversity assignment techniques are sharing or crowding. The notion of *sharing* (or *fitness sharing*) has initially been suggested by Goldberg and Richardson [30] to preserve diversity among the solutions of an EA population. It has first been employed by Fonseca and Fleming [5] in the frame of EMO. This *kernel* method consists of 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 threshold, *i.e.* the size of *niches*. The distance measure between two solutions can be defined in the decision space, in the objective space or can even combined both. Nevertheless, a distance metric partly or fully defined in the parameter space strongly depends of the tackled problem. Another diversity assignment scheme is the concept of *crowding*, firstly suggested by Holland [31] and used by De Jong to prevent *genetic drift* [32]. It is employed by Deb et al. [7] in the frame of NSGA-II. Contrary to sharing, this scheme allows to maintain diversity without specifying any parameter. It consists in estimating the density of solutions surrounding a particular point of the objective space.

2.1.7 Selection

The selection step is one of the main search operators of EAs. It consists of choosing some solutions that will be used to generate the offspring population. In general, the better is an individual, the higher is its chance of being selected. Common strategies are deterministic or stochastic tournament, roulette-wheel selection, random selection, etc. An existing EMO-specific elitist scheme consists of including solutions from the archive in the selection process, so that nondominated solutions also contributes to the evolution engine. Such an approach has successfully been applied in various elitist EMO algorithms including SPEA [9], SPEA2 [10] or PESA [11]. In addition, in order to prohibit the crossover of dissimilar parents, mating restriction [26] can also be mentioned as a candidate strategy to be integrated into EMO algorithms.

2.1.8 Replacement

Selection pressure is also affected at the replacement step where survivors are selected from both the current and the offspring population. In generational replacement, the offspring population systematically replace the parent one. An elitist strategy consists of selecting the N best solutions from both populations, where N stands for the appropriate population size.

2.1.9 Elitism

Another essential issue about MOP solving is the notion of *elitism*. It mainly consists of maintaining an external set, the so-called *archive*, that allows to store either all or a subset of nondominated solutions found during the search process. This secondary population mainly aims at preventing the loss of these solutions during the stochastic optimization process. The update of the archive contents with new potential nondominated solutions is mostly based on the Pareto-dominance criteria. But, in the literature, other dominance criterion are found and can be used instead of the Pareto-dominance relation. Examples are weak-dominance, strict-dominance, ϵ-dominance [33], etc. When dealing about archiving, one may distinguished four different techniques depending on the problem properties, the designed algorithm and the number of desired solutions: (*i*) *no archive*, (*ii*) an *unbounded archive*, (*iii*) a *bounded archive* or (*iv*) a *fixed-size archive*. Firstly, if the current approximation is maintained by, or contained in the main population itself, there can be no archive at all. On the other hand, if an archive is maintained, it usually comprises the current nondominated set approximation, as dominated solutions are removed. Then, an unbounded archive can be used in order to save the whole set of nondominated solutions found until the beginning of the search process. However, as some continuous optimization problems may contain an infinite number of nondominated solutions, it is simply not possible to save them all. Therefore, additional operations must be used to reduce the number of stored solutions. Then, a common strategy is to bound the size of the archive according to some fitness and/or diversity assignment scheme(s). Finally, another archiving technique consists of a fixed size storage capacity, where a bounding mechanism is used when there is too much nondominated solutions, and some dominated solutions are integrated in the archive if the nondominated set is too small, what is done for instance in SPEA2 [10]. Usually, an archive is used as an external storage only. However, archive members can also be integrated during the selection phase of an EMO algorithm [9], see Sect. 2.1.7.

2.1.10 Stopping criteria

Since an iterative method computes successive approximations, a practical test is required to determine when the process must stop. Popular examples are a given number of iterations, a given number of evaluations, a given run time, etc.

2.2 State-of-the-art EMO Methods as Instances of the Proposed Model

By means of the unified model proposed in this paper, we claim that a large number of state-of-the-art EMO algorithms proposed in the last two decades are

based on variations of the problem-independent components presented above. In Table 1, three EMO approaches, namely NSGA-II [7], SPEA2 [10] and IBEA [22], are regarded as simple instances of the unified model proposed in this paper. Of course, only problem-independent components are presented. NSGA-II and SPEA2 are two of the most frequently encountered EMO algorithms of the literature, either for tackling an original MOP or to serve as references for comparison. Regarding IBEA, it is a good illustration of the new EMO trend dealing with indicator-based search that started to become popular in recent years. We can see in the table that these three state-of-the-art algorithms perfectly fit into our unified model for EMO, what strongly validates the proposed approach. But other examples can be found in the literature. For instance, the only components that differ from NSGA [6] to NSGA-II [7] is the diversity preservation strategy, that is based on sharing in NSGA and on crowding in NSGA-II. Another example is the ϵ-MOEA proposed in [27]. This algorithm is a modified version of NSGA-II where the Pareto-dominance relation used for fitness assignment has been replaced by the ϵ-dominance relation. Similarly, the g-dominance relation proposed in [28] is experimented by the authors on a NSGA-II-like EMO technique where the dominance relation has been modified in order to take the DM preferences into account by means of a reference point.

3 Design and Implementation under ParadisEO-MOEO

In this section, we provide a general presentation of ParadisEO, a software framework dedicated to the design of metaheuristics, and a detailed description of the ParadisEO module specifically dedicated to EMO, namely ParadisEO-MOEO. Historically, ParadisEO was especially dedicated to parallel and distributed metaheuristics and was the result of the PhD work of Sébastien Cahon, supervised by Nouredine Melab and El-Ghazali Talbi [34]. The initial version already contained a few number of EMO-related features, mainly with regard to archiving. This work has been partially extended and presented in [35]. But since then, the ParadisEO-MOEO module has been completely redesigned in order to confer an even more fine-grained decomposition in accordance with the unified model presented above.

3.1 Motivations

In practice, there exists a large diversity of optimization problems to be solved, engendering a wide number of possible models to handle in the frame of a metaheuristic solution method. Moreover, a growing number of general-purpose search methods are proposed in the literature, with evolving complex mechanisms. From a practitioner point of view, there is a popular demand to provide a set of ready-to-use metaheuristic implementations, allowing a minimum programming effort. On the other hand, an expert generally wants to be able to design new algorithms, to integrate new components into an existing method, or even to combine different search mechanisms. As a consequence, an approved approach for the development of metaheuristics is the use of frameworks. A metaheuristic software framework may be defined by a set of components based

Table 1: State-of-the-art EMO methods as instances of the proposed unified model.

Components	*NSGA-II* [7]	*SPEA2* [10]	*IBEA* [22]
Fitness assignment	dominance-depth	dominance-count and rank	binary quality indicator
Diversity assignment	crowding distance	nearest neighbor	none
Selection	binary tournament	binary tournament	binary tournament
Replacement	elitist replacement	generational replacement	elitist replacement
Archiving	none	fixed-size archive	none
Stopping criteria	max. number of generations	max. number of generations	max. number of generations

on a strong conceptual separation of the invariant part and the problem-specific part of metaheuristics. Then, each time a new optimization problem is tackled, both code and design can directly be reused in order to redo as little code as possible.

3.2 ParadisEO and ParadisEO-MOEO

ParadisEO[2] is a white-box object-oriented software framework dedicated to the flexible design of metaheuristics for optimization problems of both discrete and combinatorial nature. Based on EO (Evolving Objects)[3] [36], this template-based, ANSI-C++ compliant computation library is portable across both Unix-like and Windows systems. Moreover, it tends to be used both by non-specialists and optimization experts. ParadisEO is composed of four connected modules that constitute a global framework. Each module is based on a clear conceptual separation of the solution methods from the problems they are intended to solve. This separation confers a maximum code and design reuse to the user. The first module, ParadisEO-EO, provides a broad range of components for the development of population-based metaheuristics, including evolutionary algorithms or particle swarm optimization techniques. Second, ParadisEO-MO contains a set of tools for single-solution based metaheuristics, *i.e.* local search, simulated annealing, tabu search, etc. Next, ParadisEO-MOEO is specifically dedicated to the reusable design of metaheuristics for multiobjective optimization. Finally, ParadisEO-PEO provides a powerful set of classes for the design of parallel and distributed metaheuristics: parallel evaluation of solutions, parallel evaluation function, island model and cellular model. In the frame of this paper, we will exclusively focus on the module devoted to multiobjective optimization, namely ParadisEO-MOEO.

ParadisEO-MOEO provides a flexible and modular framework for the design of EMO metaheuristics. Its implementation is based on the unified model proposed in the previous section and 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 instantiable classes, corresponding to different state-of-the-art strategies, are also provided. Moreover, as the framework aims to be extensible, flexible and easily adaptable, all its components are generic so that its modular architecture allows to quickly and conveniently develop any new scheme with a minimum code writing. The underlying goal here is to follow new strategies coming from the literature and, if need be, to provide any additional components required for their implementation. ParadisEO-MOEO constantly evolves and new features might be regularly added to the framework in order to provide a wide range of efficient and modern concepts and to reflect the most recent advances of the EMO field.

3.3 Main Characteristics

A framework is usually intended to be exploited by a large number of users. Its exploitation could only be successful if a range of user criteria are satisfied. Therefore, the main goals of the ParadisEO software framework are the following ones:

[2] `http://paradiseo.gforge.inria.fr`
[3] `http://eodev.sourceforge.net`

- *Maximum design and code reuse.* The framework must provide a whole architecture design for the metaheuristic approach to be used. Moreover, the programmer may redo as little code as possible. This aim requires a clear and maximal conceptual separation of the solution methods and the problem to be solved. The user might only write the minimal problem-specific code and the development process might be done in an incremental way, what will considerably simplify the implementation and reduce the development time and cost.

- *Flexibility and adaptability.* It must be possible to easily add new features or to modify existing ones without involving other components. Users must have access to source code and use inheritance or specialization concepts of object-oriented programming to derive new components from base or abstract classes. Furthermore, as existing problems evolve and new others arise, the framework components must be conveniently specialized and adapted.

- *Utility.* The framework must cover a broad range of metaheuristics, fine-grained components, problems, parallel and distributed models, hybridization mechanisms, etc.

- *Transparent and easy access to performance and robustness.* As the optimization applications are often time-consuming, the performance issue is crucial. Parallelism and distribution are two important ways to achieve high performance execution. Moreover, the execution of the algorithms must be robust in order to guarantee the reliability and the quality of the results. Hybridization mechanisms generally allow to obtain robust and better solutions.

- *Portability.* In order to satisfy a large number of users, the framework must support many material architectures (sequential, parallel, distributed) and their associated operating systems (Windows, Linux, MacOS).

- *Usability and efficiency.* The framework must be easy to use and must not contain any additional cost in terms of time or space complexity in order to keep the efficiency of a special-purpose implementation. On the contrary, the framework is intented to be less error-prone than a specifically developed metaheuristic.

The ParadisEO platform honors all the above-mentioned criteria and aims to be used by both non-specialists and optimization experts. Furthermore, The ParadisEO-MOEO module must cover additional goals related to EMO. Thus, in terms of design, it might for instance be a commonplace to extend a single-objective optimization problem to the multiobjective case without modifying the whole metaheuristic implementation.

3.4 Existing Software Frameworks for Evolutionary Multiobjective Optimization

Many frameworks dedicated to the design of metaheuristics have been proposed so far. However, very few are able to handle MOPs, even if some of them provide components for a few particular EMO strategies, such as ECJ [37], JavaEVA [38]

or Open BEAGLE [39]. Table 2 gives a non-exhaustive comparison between a number of existing software frameworks for EMO, including jMetal [18], the MOEA toolbox for Matlab a [19], MOMHLib++ [20], PISA [17] and Shark [21]. Note that other software packages exist for multiobjective optimization [40], but some cannot be considered as frameworks and others do not deal with EMO. The frameworks presented in Table 2 are distinguished according to the following criteria: the kind of MOPs they are able to tackle (continuous and/or combinatorial problems), the availability of statistical tools (including performance metrics), the availability of hybridization or parallel features, the framework type (black box or white box), the programming language and the license type (free or commercial). Firstly, let us mentioned that every listed software framework is free of use, except the MOEA toolbox designed for the commercial-software Matlab. They can all handle continuous problem, but only a subpart is able to deal with combinatorial MOPs. Moreover, some cannot be considered as white-box frameworks since their architecture is not decomposed into components. For instance, to design a new algorithm under PISA, it is necessary to implement it from scratch, as no existing element can be reused. Similarly, even if Shark can be considered as a white-box framework, its components are not as fine-grained as the ones of ParadisEO. On the contrary, ParadisEO is an open platform where anyone can contribute and add his/her own features. Finally, only a few ones are able to deal with hybrid and parallel metaheuristics at the same time. Hence, with regards to the taxonomy proposed in [41], only relay hybrid metaheuristics can be easily implemented within jMetal, MOMHLib++ and Shark, whereas ParadisEO provides tools for the design of all classes of hybrid models, including teamwork hybridization. Furthermore, in opposition to jMetal and MOMHLib++, ParadisEO offers easy-to-use models for the design of parallel and distributed EMO algorithms. Therefore, ParadisEO seems to be the only existing software framework that achieves all the aforementioned goals.

3.5 Implementation

This section gives a detailed description of the base classes provided within the ParadisEO framework to design an EMO algorithm[4]. The flexibility of the framework and its modular architecture based on the three main multiobjective metaheuristic design issues (fitness assignment, diversity preservation and elitism) allows to implement efficient algorithms in solving a large diversity of MOPs. The granular decomposition of ParadisEO-MOEO is based on the unified model proposed in the previous section.

As an EMO algorithm differs of a mono-objective one only in a number of points, some ParadisEO-EO components are directly reusable in the frame of ParadisEO-MOEO. Therefore, in the following, note that the names of ParadisEO-EO classes are all prefixed by `eo` whereas the names of ParadisEO-MOEO classes are prefixed by `moeo`. ParadisEO is an object-oriented platform, so that its components will be specified by the UML standard [42]. But, due to space limitation, only a subpart of UML diagrams are given, but the whole inheritance diagram as well as class documentation and many examples of use are available on the ParadisEO website. Moreover, a large part of ParadisEO components are based on the notion of *template* and are defined as class templates. This

[4]The classes presented in this paper are described as in version 1.2 of ParadisEO.

Table 2: Main characteristics of some existing frameworks for multiobjective metaheuristics.

Framework	Problems		Statistical tools		Hybrid.	Parallel	Type	Lang.	License
	Cont.	Comb.	Off-line	On-line					
jMetal	yes	yes	yes	no	yes	no	white	java	free
MOEA for Matlab	yes	no	no	no	no	yes	black	matlab	free / com.
MOMHLib++	yes	yes	no	no	yes	no	white	c++	free
PISA	yes	yes	yes	no	no	no	black	any	free
Shark	yes	no	no	no	yes	no	white	c++	free
ParadisEO	yes	yes	yes	yes	yes	yes	white	c++	free

concept and many related functions are featured within the C++ programming language and allows classes to handle generic types, so that they can work with many different data types without having to be rewritten for each one.

In the following, both problem-dependent and problem-independent components are detailed. Hence, basic (representation, evaluation, initialization and stopping criteria), EMO-specific (fitness, diversity and elitism) and EA-related (variation, selection, replacement) components are outlined. Finally, the way to build a whole EMO algorithm is presented and a brief discussion concludes the section.

3.5.1 Representation

A solution needs to be represented both in the decision space and in the objective space. While the representation in the objective space can be seen as problem-independent, the representation in the decision space must be relevant to the tackled problem. Using ParadisEO-MOEO, the first thing to do is to set the number of objectives for the problem under consideration and, for each one, if it has to be minimized or maximized. Then, a class inheriting of `moeoObjectiveVector` has to be created for the representation of an objective vector, as illustrated in Fig. 1. Besides, as a big majority of MOPs deals with real-coded objective values, a class modeling real-coded objective vectors is already provided. Note that this class can also be used for any MOP without loss of generality. Next, the class used to represent a solution within ParadisEO-

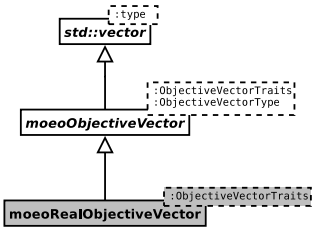

Figure 1: UML diagram for the representation of a solution in the objective space.

MOEO must 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. Nevertheless, ParadisEO-MOEO also provides easy-to-use classes for standard vector-based representations and, in particular, implementations for vectors composed of bits, of integers or of real-coded values that can thus directly be used in a ParadisEO-MOEO-designed application. These classes are summarized in Fig. 2.

3.5.2 Initialization

A number of initialization schemes already exists in a lot of libraries for standard representations, what is also the case within ParadisEO. But some situations could require a combination of many operators or a specific implementation. Indeed, the framework provides a range of initializers all inheriting of `eoInit`, as well as an easy way to combine them thanks to a `eoCombinedInit` object.

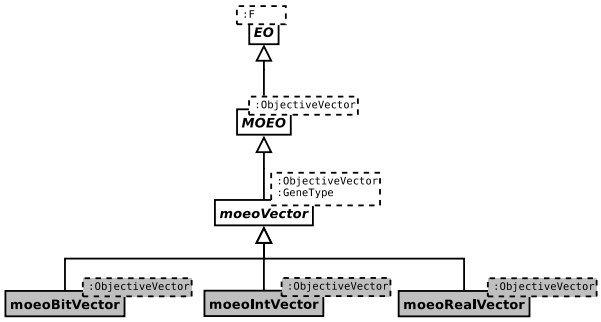

Figure 2: UML diagram for the representation of a solution.

3.5.3 Evaluation

The way to evaluate a given solution must be ensured by components inheriting of the `eoEvalFunc` abstract class. It basically takes a `MOEO` object and sets its objective vector. Generally speaking, for real-world optimization problems, evaluating a solution in the objective space is by far the most computationally expensive step of any metaheuristic. A possible way to overcome this trouble is the use of parallel and distributed models, that can largely be simplify in the frame of ParadisEO thanks to the ParadisEO-PEO module of the software library. The reader is referred to [34] for more information on how to parallelize the evaluation step of a metaheuristic within ParadisEO-PEO.

3.5.4 Variation

All variation operators must derive from the `eoOp` base class. Four abstract classes inherit of `eoOp`, namely `eoMonOp` for mutation operators, `eoBinOp` and `eoQuadOp` for recombination operators and `eoGenOp` for other kinds of variation operators. Various operators of same arity can also be combined using some helper classes. Note that variation mechanisms for some classical (real-coded, vector-based or permutation-based) representations are already provided in the framework. Moreover, an hybrid mechanism can easily be designed by using a mono-objective local search as mutation operator, as they both inherit of the same class, see ParadisEO-MO [43]. The set of all variation operators designed for a given problem must be embedded into a `eoTranform` object.

3.5.5 Fitness Assignment

Following the taxonomy introduced in Sect. 2, the fitness assignment schemes are classified into four main categories, as illustrated in the UML diagram of Fig. 3: scalar approaches, criterion-based approaches, dominance-based approaches and indicator-based approaches. Non-abstract fitness assignment schemes provided within ParadisEO-MOEO are the *achievement scalarizing functions*, the *dominance-rank*, *dominance-count* and *dominance-depth* schemes, as well as the *indicator-based* fitness assignment strategy proposed in [22]. Moreover, a dummy fitness assignment strategy has been added in case it would be useful for some specific implementation.

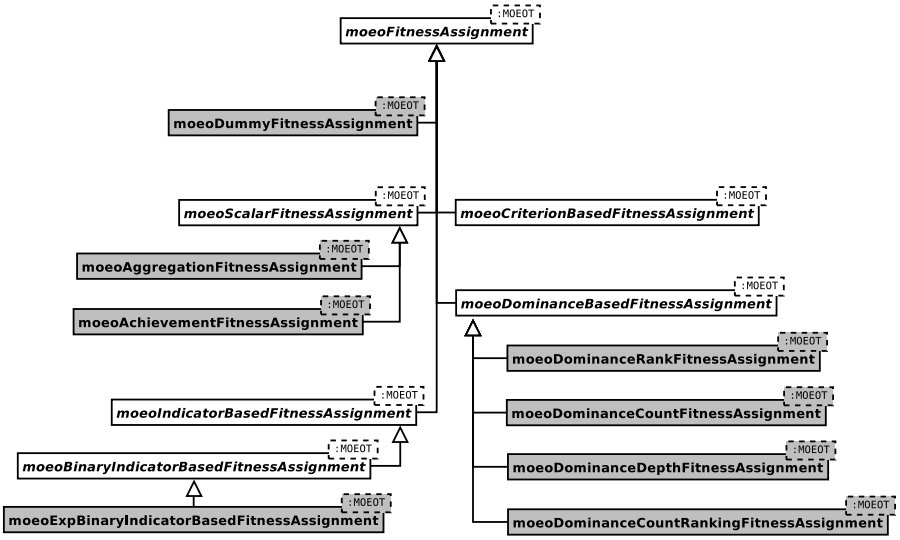

Figure 3: UML diagram for fitness assignment.

3.5.6 Diversity Assignment

As illustrated in Fig. 4, the diversity preservation strategy to be used must inherit of the `moeoDiversityAssignment` class. Hence, in addition to a dummy technique, a number diversity assignment schemes are already available, including sharing [30], crowding [31] and a nearest neighbor scheme [10].

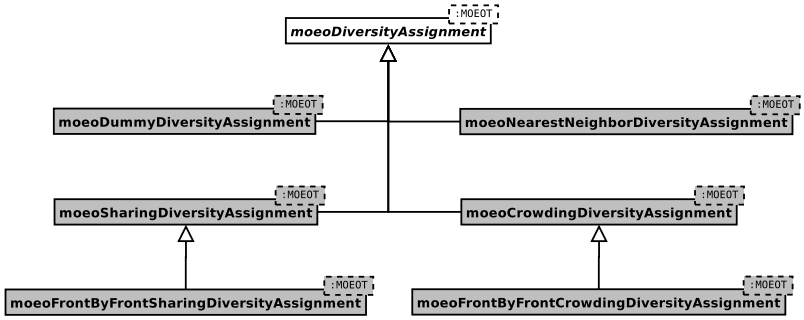

Figure 4: UML diagram for diversity assignment.

3.5.7 Selection

There exists a large number of selection strategies in the frame of EMO. Four ones are provided within ParadisEO-MOEO. First, the *random selection* consists of selecting a parent randomly among the population members, without taking nor the fitness or the diversity information into account. Second, the *deterministic tournament selection* consists of performing a tournament between m randomly chosen population members and in selecting the best one. Next, the *stochastic tournament selection* consists of performing a binary tournament

between randomly chosen population members and in selecting the best one with a probability p or the worst one with a probability $(1-p)$. Finally, the *elitist selection* consists of selecting a population member based on some selection scheme with a probability p, or in selecting an archive member using another selection scheme with a probability $(1-p)$. Thus, nondominated (or most-preferred) solutions also contribute to the evolution engine by being used as parents. A selection method needs to be embedded into a `eoSelect` object to be properly used. Of course, everything is done to easily implement a new selection scheme with a minimum programming effort.

3.5.8 Replacement

A large majority of replacement strategies depends on the fitness and/or the diversity value(s) and can then be seen as EMO-specific. Three replacement schemes are provided within ParadisEO-MOEO, but this list is not exhaustive as new ones can easily be implemented due to the genericity of the framework. First, the *generational replacement* consists of keeping the offspring population only, while all parents are deleted. Next, the *one-shot elitist replacement* consists of preserving the N best solutions, where N stands for the population size. At last, the *iterative elitist replacement* consists of repeatedly removing the worst solution until the required population size is reached. Fitness and diversity information of remaining individuals is updated each time there is a deletion.

3.5.9 Elitism

As shown in Fig. 5, in terms of implementation, an archive is represented by the `moeoArchive` abstract class and is a population using a particular dominance relation to update its contents. An abstract class for fixed-size archives is given, but implementations of an unbounded archive, a general-purpose bounded archive based on a fitness and/or a diversity assignment scheme(s) as well as the SPEA2 archive are provided. Furthermore, as shown in Fig. 6, ParadisEO-MOEO offers the opportunity to use different dominance relation to update an archive contents by means of a `moeoObjectiveVectorComparator` object, including Pareto-dominance, weak-dominance, strict-dominance, ϵ-dominance [33], and g-dominance [28].

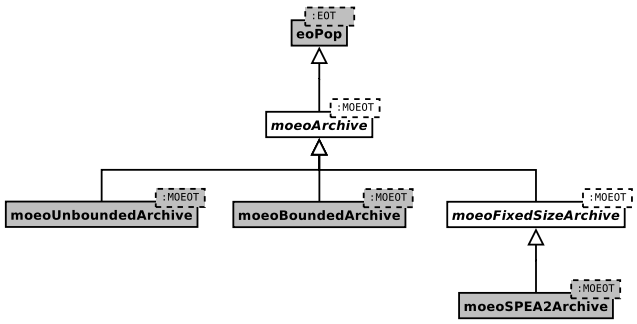

Figure 5: UML diagram for archiving.

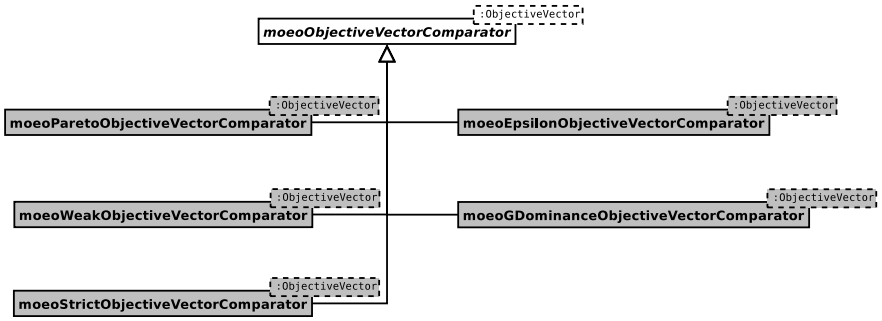

Figure 6: UML diagram for for dominance relation (used for pairwise objective vector comparison).

3.5.10 Stopping Criteria, Checkpointing and Statistical Tools

In the frame of ParadisEO, many stopping criteria extending `eoContinue` are provided. For instance, the algorithm can stop after a given number of iterations, a given number of evaluations, a given run time or in an interactive way as soon as the user decides to. Moreover, different stopping criterion can be combined, in which case the process stops once one of the embedded criteria is satisfied. In addition, many other procedures may be called at each iteration of the main algorithm. The `eoCheckPoint` class allows to perform some systematic actions at each algorithm iteration in a transparent way by being integrated into the global `eoContinue` object. The checkpointing engine is particularly helpful for fault tolerance mechanisms and to compute statistical tools. Indeed, some statistical tools are also provided within ParadisEO-MOEO. Then, it is for instance possible to save the contents of the current approximation set at each iteration, so that the evolution of the current nondominated front can be observed or study using graphical tools such as GUIMOO[5]. Furthermore, an important issue in the EMO field relates to the algorithm performance analysis and to set quality metrics [12]. A couple of metrics are featured within ParadisEO-MOEO, including the hypervolume metric in both its unary [9] and its binary [12] form, the entropy metric [44], the contribution metric [45] as well as the additive and the multiplicative ϵ-indicators [12]. Another interesting feature is the possibility to compare the current archive with the archive of the previous generation by means of a binary metric, and to print the progression of this measure iteration after iteration.

3.5.11 EMO Algorithms

Now that all the basic, EA-related and EMO-specific components are defined, an EMO algorithm can easily be designed using the fine-grained classes of ParadisEO. As the implementation is conceptually divided into components, different operators can be experimented without engendering significant modifications in terms 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

[5]GUIMOO is a Graphical User Interface for Multiobjective Optimization available at `http://guimoo.gforge.inria.fr/`

minimum effort. Indeed, ParadisEO is a white-box framework that tends to be flexible while being as user-friendly as possible. Fig. 7 illustrates the use of the `moeoEasyEA` class that allows to define an EMO algorithm in a common fashion, by specifying all the particular components required for its implementation. All classes use a template parameter *MOEOT (Multiobjective Evolving Object Type)* that defines the representation of a solution for the problem under consideration. This representation might be implemented by inheriting of the `MOEO` class as described in Sect. 3.5.1. Note that the archive-related components does not appear in the UML diagram, as we chose to let the use of an archive as optional. The archive update can easily be integrated into the EA by means of the checkpointing process. Similarly, the initialization process does not appear either, as an instance of `moeoEasyEA` starts with an already initialized population.

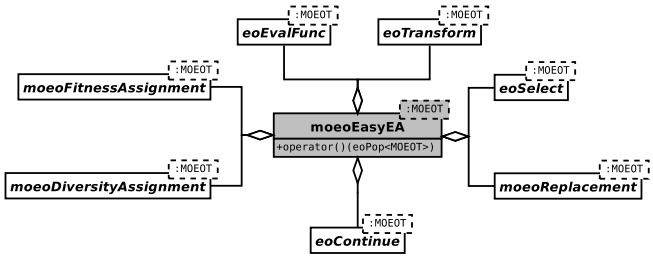

Figure 7: A possible instantiation for the design of an EMO algorithm.

In order to satisfy both the common user and the more experimented one, ParadisEO-MOEO also provides even more easy-to-use EMO algorithms, see Fig. 8. These classes propose different implementations of some state-of-the-art methods by using the fine-grained components of ParadisEO. They are based on a simple combination of components, as described in Sect. 2.2. Hence, MOGA [5], NSGA [6], NSGA-II [7], SPEA2 [10], IBEA [22] and SEEA [46] are proposed in a way that a minimum number of problem- or algorithm-specific parameters are required. For instance, to instantiate NSGA-II for a new continuous MOP, it is possible to use standard operators for representation, initialization and variation, so that the evaluation is the single component to be implemented. These easy-to-use algorithms also tends to be used as references for a fair performance comparison in the academic world, even if they are also well-suited for a straight use to solve real-world MOPs. In a near future, other easy-to-use EMO metaheuristics will be proposed while new fined-grained components will be implemented into ParadisEO-MOEO.

3.6 Discussion

ParadisEO-MOEO has been used and experimented to solve a large range of MOPs from both academic and real-world fields, what validates its high flexibility. Indeed, various academic MOPs have been tackled within ParadisEO-MOEO, including continuous test functions (like the ZDT and DTLZ functions family defined in [47]), scheduling problems (permutation flow-shop scheduling problem [48]), routing problems (multiobjective traveling salesman problem, bi-objective ring star problem [46]), and so on. Moreover, it has been successfully employed to solve real-world applications in structural biology [49], feature selec-

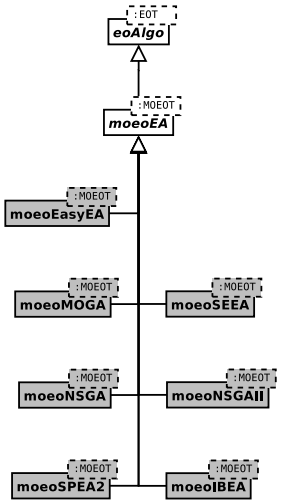

Figure 8: UML diagram for easy-to-use EMO algorithms.

tion in cancer classification [50], materials design in chemistry [51], etc. Besides, a detailed documentation as well as some tutorial lessons and problem-specific implementations are freely available on the ParadisEO website[6]. And we expect the number of MOP contributions to largely grow in a near future. Furthermore, note that the implementation of EMO algorithms is just an aspect of the features provided by ParadisEO. Hence, hybrid mechanisms can be exploited in a natural way to make cooperating metaheuristics belonging to the same or to different classes. Moreover, the three main parallel models are concerned: algorithmic-level, iteration-level and solution-level and are portable on different types of architecture. Indeed, the whole framework allows to conveniently design hybrid as well as parallel and distributed metaheuristics, including EMO methods. For instance, in the frame of ParadisEO, hybrid EMO algorithms have been experimented in [46], a multiobjective cooperative island model has been designed in [52], and costly evaluation functions have been parallelized in [49]. The reader is referred to [34] for more information about ParadisEO hybrid and parallel models.

4 Concluding Remarks

The current paper presents two complementary contributions: the formulation of a unified view for evolutionary multiobjective optimization and the description of a software framework for the development of such algorithms. First, we identified the common concepts shared by many evolutionary multiobjective optimization techniques, separating the problem-specific part from the invariant part involved in this class of resolution methods. We emphasized the main issues of fitness assignment, diversity preservation and elitism. Therefore, we proposed a unified conceptual model, based on a fine-grained decomposition, and we illustrated its robustness and its reliability by treating a number of

[6] `http://paradiseo.gforge.inria.fr`

state-of-the-art algorithms as simple instances of the model. Next, this unified view has been used as a starting point for the design and the implementation of a general-purpose software package called ParadisEO-MOEO. ParadisEO-MOEO is a free C++ white-box object-oriented framework dedicated to the flexible and reusable design of evolutionary multiobjective optimization algorithms. It is based on a clear conceptual separation between the resolution methods and the problem they are intended to solve, thus conferring a maximum code and design reuse. This global framework has been experimentally validated by solving a comprehensive number of both academic and real-world multiobjective optimization problems.

However, we believe that a large number of components involved in evolutionary multiobjective optimization are shared by many other search techniques. Thereafter, we plan to generalize the unified model proposed in this paper to other existing metaheuristic approaches for multiobjective optimization. Hence, multiobjective local search or scatter search methods might be interesting extensions to explore in order to investigate their ability and their modularity for providing such a flexible model as the one presented in this paper. Afterwards, the resulting general-purpose models and their particular mechanisms would be integrated into the ParadisEO-MOEO software framework.

Acknowledgment

The authors would like to gratefully acknowledge Thomas Legrand, Jérémie Humeau, and Abdel-Hakim Deneche for their helpful contribution on the implementation part of this work, as well as Sébastien Cahon and Nouredine Melab for their work on the preliminary version of the ParadisEO-MOEO software framework presented in this paper. This work was supported by the ANR DOCK project.

References

[1] K. Deb, *Multi-Objective Optimization using Evolutionary Algorithms*. Chichester, UK: John Wiley & Sons, 2001.

[2] C. A. Coello Coello, G. B. Lamont, and D. A. Van Veldhuizen, *Evolutionary Algorithms for Solving Multi-Objective Problems (2nd edition)*. New York, USA: Springer, 2007.

[3] E. Zitzler, M. Laumanns, and S. Bleuler, "A tutorial on evolutionary multiobjective optimization," in *Metaheuristics for Multiobjective Optimisation*, ser. Lecture Notes in Economics and Mathematical Systems, X. Gandibleux, M. Sevaux, and K. Swrensen, Eds., vol. 535. Springer-Verlag, 2004, pp. 3–38.

[4] J. D. Schaffer, "Multiple objective optimization with vector evaluated genetic algorithms," in *Proceedings of the 1st International Conference on Genetic Algorithms (ICGA 1985)*, J. J. Grefensette, Ed. Pittsburgh, PA, USA: Lawrence Erlbaum Associates, 1985, pp. 93–100.

[5] C. M. Fonseca and P. J. Fleming, "Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization," in *Proceedings of the 5th International Conference on Genetic Algorithms (ICGA 1993)*, S. Forrest, Ed. Urbana-Champaign, IL, USA: Morgan Kaufmann, 1993, pp. 416–423.

[6] N. Srinivas and K. Deb, "Multiobjective optimization using nondominated sorting in genetic algorithms," *Evolutionary Computation*, vol. 2, no. 3, pp. 221–248, 1994.

[7] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[8] J. Horn, N. Nafpliotis, and D. E. Goldberg, "A niched pareto genetic algorithm for multiobjective optimization," in *IEEE Congress on Evolutionary Computation (CEC 1994)*. Piscataway, NJ, USA: IEEE Press, 1994, pp. 82–87.

[9] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 4, pp. 257–271, 1999.

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

[11] D. Corne, J. D. Knowles, and M. J. Oates, "The pareto envelope-based selection algorithm for multi-objective optimisation," in *Conference on Parallel Problem Solving from Nature (PPSN VI)*, ser. Lecture Notes in Computer Science, vol. 1917. London, UK: Springer-Verlag, 2000, pp. 839–848.

[12] 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*, vol. 7, no. 2, pp. 117–132, 2003.

[13] M. P. Fourman, "Compaction of symbolic layout using genetic algorithms," in *Proceedings of the 1st International Conference on Genetic Algorithms (ICGA 1985)*, J. J. Grefensette, Ed. Pittsburgh, PA, USA: Lawrence Erlbaum Associates, 1985, pp. 141–153.

[14] J. D. Knowles and D. Corne, "Approximating the nondominated front using the pareto archived evolution strategy," *Evolutionary Computation*, vol. 8, no. 2, pp. 149–172, 2000.

[15] M. Laumanns, E. Zitzler, and L. Thiele, "A unified model for multiobjective evolutionary algorithms with elitism," in *IEEE Congress on Evolutionary Computation (CEC 2000)*. Piscataway, New Jersey, USA: IEEE Press, 2000, pp. 46–53.

[16] K. Deb, "A robust evolutionary framework for multi-objective optimization," in *Genetic and Evolutionary Computation Conference (GECCO 2008)*. Atlanta, GA, USA: ACM, 2008, pp. 633–640.

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

[18] J. J. Durillo, A. J. Nebro, F. Luna, B. Dorrosoro, and E. Alba, "jMetal: A java framework for developing multi-objective optimization metaheuristics," University of Málaga, Tech. Rep. ITI-2006-10, 2006.

[19] 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)*, vol. 31, no. 4, pp. 537–556, 2001.

[20] [Online]. Available: http://home.gna.org/momh/

[21] [Online]. Available: http://shark-project.sourceforge.net/

[22] E. Zitzler and S. Künzli, "Indicator-based selection in multiobjective search," in *Conference on Parallel Problem Solving from Nature (PPSN VIII)*, ser. Lecture Notes in Computer Science, vol. 3242. Birmingham, UK: Springer-Verlag, 2004, pp. 832–842.

[23] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*. New York, USA: Springer, 2003.

[24] E.-G. Talbi, *Metaheuristics: from design to implementation*. Wiley, 2009.

[25] K. Miettinen, *Nonlinear Multiobjective Optimization*, ser. International Series in Operations Research and Management Science. Boston, MA, USA: Kluwer Academic Publishers, 1999, vol. 12.

[26] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Boston, MA, USA: Addison-Wesley, 1989.

[27] K. Deb, M. Mohan, and S. Mishra, "Evaluating the ϵ-domination based multi-objective evolutionary algorithm for a quick computation of pareto-optimal solutions," *Evolutionary Computation*, vol. 13, no. 4, pp. 501–525, 2005.

[28] J. Molina, L. V. Santana, A. G. Hernández-Díaz, C. A. Coello Coello, and R. Caballero, "g-dominance: Reference point based dominance for multiobjective metaheuristics," *European Journal of Operational Research*, 2008, (in press).

[29] N. Beume, B. Naujoks, and M. Emmerich, "SMS-EMOA: Multiobjective selection based on dominated hypervolume," *European Journal of Operational Research*, vol. 181, no. 3, pp. 1653–1669, 2007.

[30] D. E. Goldberg and J. Richardson, "Genetic algorithms with sharing for multimodal function optimization," in *Second International Conference on Genetic Algorithms and their application*. Mahwah, NJ, USA: Lawrence Erlbaum Associates, Inc., 1987, pp. 41–49.

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

[32] K. A. D. Jong, "An analysis of the behavior of a class of genetic adaptive systems," Ph.D. dissertation, Ann Arbor, University of Michigan, 1975.

[33] S. Helbig and D. Pateva, "On several concepts for ϵ-efficiency," *OR Spektrum*, vol. 16, no. 3, pp. 179–186, 1994.

[34] S. Cahon, N. Melab, and E.-G. Talbi, "ParadisEO: A framework for the reusable design of parallel and distributed metaheuristics," *Journal of Heuristics*, vol. 10, no. 3, pp. 357–380, 2004.

[35] A. Liefooghe, M. Basseur, L. Jourdan, and E.-G. Talbi, "ParadisEO-MOEO: A framework for evolutionary multi-objective optimization," in *Evolutionary Multi-Criterion Optimization, Fourth International Conference (EMO 2007)*, ser. Lecture Notes in Computer Science, S. Obayashi, C. Poloni, and K. Deb, Eds., vol. 4403. Matsushima, Japan: Springer-Verlag, 2007, pp. 386–400.

[36] 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)*, Le Creusot, France, 2001, pp. 231–244.

[37] [Online]. Available: http://cs.gmu.edu/~eclab/projects/ecj/

[38] F. Streichert and H. Ulmer, "JavaEvA : a java based framework for evolutionary algorithms," Centre for Bioinformatics Tübingen (ZBIT) of the Eberhard-Karls-University, Tübingen, Tech. Rep. WSI-2005-06, 2005.

[39] C. Gagné and M. Parizeau, "Genericity in evolutionary computation software tools: Principles and case study," *International Journal on Artificial Intelligence Tools*, vol. 15, no. 2, pp. 173–194, 2006.

[40] S. Poles, M. Vassileva, and D. Sasaki, "Multiobjective optimization software," in *Multiobjective Optimization - Interactive and Evolutionary Approaches*, ser. Lecture Notes in Computer Science (LNCS), J. Branke, K. Deb, K. Miettinen, and R. Slowínski, Eds. Berlin Heidelberg: Springer-Verlag, 2008, vol. 5252, ch. 12, pp. 329–348.

[41] E.-G. Talbi, "A taxonomy of hybrid metaheuristics," *Journal of Heuristics*, vol. 8, no. 2, pp. 541–564, 2002.

[42] "OMG unified modeling language specification," Object Management Group, 2000.

[43] J. C. Boisson, L. Jourdan, and E.-G. Talbi, "ParadisEO-MO," Tech. Rep., 2008.

[44] M. Basseur, F. Seynhaeve, and E.-G. Talbi, "Design of multi-objective evolutionary algorithms: Application to the flow shop scheduling problem," in *IEEE Congress on Evolutionary Computation (CEC 2002)*. Piscataway, NJ, USA: IEEE Press, 2002, pp. 1151–1156.

[45] H. Meunier, E.-G. Talbi, and P. Reininger, "A multiobjective genetic algorithm for radio network optimization," in *IEEE Congress on Evolutionary Computation (CEC 2000)*. San Diego, USA: IEEE Press, 2000, pp. 317–324.

[46] A. Liefooghe, L. Jourdan, and E.-G. Talbi, "Metaheuristics and their hybridization to solve the bi-objective ring star problem: a comparative study," Institut National de Recherche en Informatique et Automatique (INRIA), Tech. Rep. RR-6515, 2008.

[47] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, "Scalable test problems for evolutionary multi-objective optimization," in *Evolutionary Multiobjective Optimization: Theoretical Advances and Applications*, A. Abraham, R. Jain, and R. Goldberg, Eds. Springer, 2005, ch. 6, pp. 105–145.

[48] A. Liefooghe, M. Basseur, L. Jourdan, and E.-G. Talbi, "Combinatorial optimization of stochastic multi-objective problems: an application to the flow-shop scheduling problem," in *Evolutionary Multi-criterion Optimization (EMO 2007)*, ser. Lecture Notes in Computer Science, S. Obayashi, K. Deb, C. Poloni, T. Hiroyasu, and T. Murata, Eds., vol. 4403. Matsushima, Japan: Springer-Verlag, 2007, pp. 457–471.

[49] J.-C. Boisson, L. Jourdan, E.-G. Talbi, and D. Horvath, "Parallel multi-objective algorithms for the molecular docking problem," in *IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB 2008)*, Sun Valley Resort, Idaho, USA, 2008.

[50] E.-G. Talbi, L. Jourdan, J. Garcia-Nieto, and E. Alba, "Comparison of population based metaheuristics for feature selection: Application to microarray data classification," in *IEEE/ACS International Conference on Computer Systems and Applications (AICCSA 2008)*. IEEE, 2008, pp. 45–52.

[51] O. Schuetze, L. Jourdan, T. Legrand, E.-G. Talbi, and J.-L. Wojkiewicz, "New analysis of the optimization of electromagnetic shielding properties using conducting polymers and a multi-objective approach," *Polymers for Advanced Technologies*, vol. 19, no. 7, pp. 762–769, 2008.

[52] E.-G. Talbi, S. Cahon, and N. Melab, "Designing cellular networks using a parallel hybrid metaheuristic on the computational grid," *Computer Communications*, vol. 30, no. 4, pp. 698–713, 2007.

Contents

1 Introduction **3**

2 The Proposed Unified Model **5**
2.1 Components Description 6
2.1.1 Representation 6
2.1.2 Initialization 6
2.1.3 Evaluation 6
2.1.4 Variation 7
2.1.5 Fitness Assignment 7
2.1.6 Diversity Assignment 8
2.1.7 Selection 8
2.1.8 Replacement 9
2.1.9 Elitism 9
2.1.10 Stopping criteria 9
2.2 State-of-the-art EMO Methods as Instances of the Proposed Model 9

3 Design and Implementation under ParadisEO-MOEO **10**
3.1 Motivations 10
3.2 ParadisEO and ParadisEO-MOEO 12
3.3 Main Characteristics 12
3.4 Existing Software Frameworks for Evolutionary Multiobjective Optimization 13
3.5 Implementation 14
3.5.1 Representation 16
3.5.2 Initialization 16
3.5.3 Evaluation 17
3.5.4 Variation 17
3.5.5 Fitness Assignment 17
3.5.6 Diversity Assignment 18
3.5.7 Selection 18
3.5.8 Replacement 19
3.5.9 Elitism 19
3.5.10 Stopping Criteria, Checkpointing and Statistical Tools . . 20
3.5.11 EMO Algorithms 20
3.6 Discussion 21

4 Concluding Remarks **22**

Centre de recherche INRIA Lille – Nord Europe
Parc Scientifique de la Haute Borne - 40, avenue Halley - 59650 Villeneuve d'Ascq (France)

Centre de recherche INRIA Bordeaux – Sud Ouest : Domaine Universitaire - 351, cours de la Libération - 33405 Talence Cedex
Centre de recherche INRIA Grenoble – Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier
Centre de recherche INRIA Nancy – Grand Est : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex
Centre de recherche INRIA Paris – Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex
Centre de recherche INRIA Rennes – Bretagne Atlantique : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex
Centre de recherche INRIA Saclay – Île-de-France : Parc Orsay Université - ZAC des Vignes : 4, rue Jacques Monod - 91893 Orsay Cedex
Centre de recherche INRIA Sophia Antipolis – Méditerranée : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399