A unified model for evolutionary multi-objective optimization and its implementation in a general purpose software framework

Arnaud Liefooghe, Laetitia Jourdan, El-Ghazali Talbi

A Unified Model for Evolutionary Multi-objective Optimization and its Implementation in a General Purpose Software Framework

Arnaud Liefooghe, Laetitia Jourdan, and El-Ghazali Talbi

Abstract— **The aim of this paper is to propose a unified view of evolutionary approaches for multi-objective optimization. Following three main issues dealing with fitness assignment, diversity preservation and elitism, a robust and flexible model, based on a fine-grained decomposition, is introduced. This model is validated by demonstrating how state-of-the-art methods can conveniently fit into it. Then, a modular implementation is proposed and is successfully integrated in a general purpose software framework dedicated to the reusable design of evolutionary multi-objective optimization techniques.**

I. Introduction

Multi-objective optimization is one of the most challenging areas in the field of multiple criteria decision making. Over the last decades, major contributions were published in this field comprising both theoretical and applied works. Contrary to the single-objective case, there does not exist a unique optimal solution for a multi-objective problem. Instead, a set of compromise solutions, known as efficient solutions, and equally good mathematically speaking, are generally identified. In recent years, a large number of approaches were devoted to the identification of this set or a good approximation of it, where evolutionary algorithms seem particularly efficient. In this paper, we propose a unified view of evolutionary algorithms for multi-objective optimization. We describe the basic components that are common to every multi-objective evolutionary algorithm, and we introduce a general purpose model as well as a classification of its fine-grained components. Next, we confirm its high genericity by treating a number of state-of-the-art methods as simple instances of the model, including NSGA-II [1], SPEA2 [2] and IBEA [3]. Then, we illustrate how this unified 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 evolutionary multi-objective optimization methods, namely ParadisEO-MOEO. This free C++ white-box framework has been widely experimented and has enabled the resolution of a large variety of real-world multi-objective optimization problems.

Arnaud Liefooghe, Laetitia Jourdan and El-Ghazali Talbi are with the LIFL, INRIA Lille-Nord Europe, Université des Sciences et Technologies de Lille, 40 avenue Halley, 59650 Villeneuve d'Ascq, France (email: {Arnaud.Liefooghe, Laetitia.Jourdan, El-Ghazali.Talbi}@lifl.fr).

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.

Note that similar attempts have been made in the past, including the unification model proposed in [4], that resulted in the PISA framework. However, we expect the unified view presented in this paper to be more complete and to provide a more fine-grained decomposition, and the software framework to provide a more modular implementation.

The outline of the paper is as follows. In Section II, a few words about evolutionary multi-objective optimization are given. The unified model is presented in details in Section III. Section IV is devoted to the design and the implementation of evolutionary multi-objective optimization metaheuristics under ParadisEO-MOEO. Then, the paper finishes with some concluding remarks.

II. Evolutionary Multi-objective Optimization Design Issues

This section presents some basic concepts about Evolutionary Multi-objective Optimization (EMO). Next, a couple of related design issues are briefly discussed.

A. Evolutionary Multi-objective Optimization

A Multi-objective Optimization Problem (MOP) can be defined by a set 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 here 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 $z = f(x) = f_1(x), f_2(x), \ldots, f_n(x)$. An objective vector $z \in Z$ is said to *dominate* (in the Pareto sense) 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 *non-dominated* if there does not exist any other objective vector $z' \in Z$ such that z' dominates z. By extension, we 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 *non-dominated* (or *efficient*, *Pareto optimal*) if $f(x)$ is non-dominated. A possible MOP resolution method is to find the minimal set of efficient solutions, *i.e.* one feasible solution per non-dominated point. However, generating the entire efficient set is usually infeasible, due to the complexity of the underlying problem or the large number of optima. Therefore, in many approaches, the overall goal is 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 [5], [6] for more details about EMO.

B. Design Issues

As pointed out by various authors (see *e.g.* [4], [6]), approximating the 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 (unknown) optimal Pareto front. As a consequence, the main differences between the design of a single- and a multi-objective metaheuristic in general, and EA in particular, deal with these two goals. As noticed by Zitzler et al. [4], in the EMO literature, initial approaches were mainly focused on moving toward the Pareto front [7]. Afterward, diversity preservation mechanisms quickly emerged [8], [9]. Then, at the end of the nineteens, the concept of elitism, related to the preservation of non-dominated solutions, became very popular and is now employed in most recent EMO methods [10], [2]. The importance of the issues of *fitness assignment*, *diversity preservation* and *elitism* are commonly approved and are also presented under different names in, for instance, [4], [6]. They are discussed in details in the next section.

III. The Proposed Unified Model

An Evolutionary Algorithm (EA) [11] is a search method 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 multi-objective 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 non-dominated 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. Following this fine-grained decomposition, a conceptual unified model for EMO is proposed and presented in Fig 1. Problem-dependent components appear in gray areas in the figure.

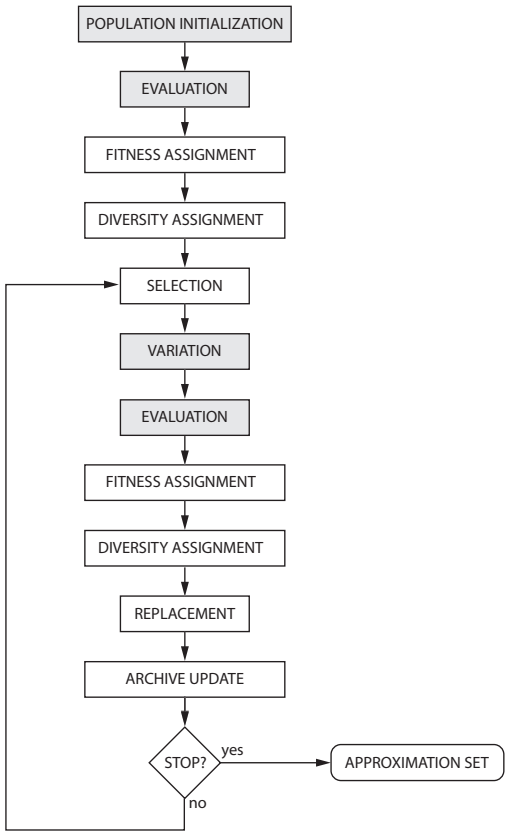

Fig. 1. Diagram of the proposed unified model for EMO

A. Components Description

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

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) 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.

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.

4) Variation: The purpose of variation operators is to modify the representation of solution 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.

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. Existing fitness assignment schemes can be classified into four different classes:

- *Scalar approaches*, where the MOP is reduced to an optimization problem with a single objective function. A popular example consists of combining the n objective functions into a single one by means of a weighted-sum aggregation. Other examples are lexicographic or achievement function-based methods [12].
- *Criterion-based approaches*, where each objective function is treated separately, like in VEGA (Vector Evaluated GA) [7].
- *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 [8]. In *dominance-count* techniques, the fitness value of a solution corresponds to the number of individuals that are dominated by these solutions [10]. Finally, *dominance-depth* strategies consists of classifying the population into different fronts [13]. Note that different schemes can also be combined, as for instance in [10]. In the frame of dominance-based approaches, the most commonly used dominance relation is the Pareto-dominance relation. But some recent techniques are based on other dominance operators such as ϵ-dominance in [14] or g-dominance in [15].
- *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) [3] or SMS-EMOA (S Metric Selection EMO Algorithm) [16].

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 non-dominated 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 EMO 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 diversity assignment techniques are *sharing* [17] or *crowding* [18], respectively used in *e.g.* Fonseca and Fleming MOGA (Multi-Objective GA) [8] and Deb et al. NSGA-II (Non-dominated Sorting GA II) [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... An existing EMO-specific elitist scheme consists of including solutions from the archive in the selection process [10], so that non-dominated solutions also contributes to the evolution engine. In addition, in order to prohibit the crossover of dis-

similar parents, mating restriction [13] can also be mentioned as a good strategy to be integrated into EMO algorithms.

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.

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 non-dominated 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 non-dominated 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 [19], etc. When dealing about archiving, one may distinguished four different techniques depending on the problem properties and on the designed algorithm: (i) *no archive*, (ii) an *unbounded archive*, (iii) a *bounded archive* or (iv) a *fixed-size archive*. First, there can be no archive at all in case of the approximation set is maintained by, or contained in the EA population itself. On the other hand, if an archive is maintained, it usually comprises the current non-dominated set approximation, as dominated solutions are removed. Then, an unbounded archive can be used in order to save the whole set of non-dominated solutions found until the beginning of the search process. However, as some continuous optimization problems may contain an infinite number of non-dominated 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. Finally, another archiving technique consists of a fixed size storage capacity, where a bounding mechanism is used when there is too much non-dominated solutions, and some dominated solutions are integrated in the archive if the non-dominated set is too small, what is done for instance in SPEA2 [2]. 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 [10].

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.

B. 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 I, three EMO approaches, namely NSGA-II [1], SPEA2 [2] and IBEA [3], are regarded as simple instances of our unified model. 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. IBEA is a more modern method that started to become popular in recent years, probably due its efficiency and its easy of use. 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 [9] to NSGA-II [1] 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 [14]. 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 [15] 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.

IV. 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 [20]. 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 [21]. 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.

A. 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 framework may be defined by a set of

TABLE I

STATE-OF-THE-ART EMO METHODS AS INSTANCES OF THE PROPOSED UNIFIED MODEL

Components	*NSGA-II* [1]	*SPEA2* [2]	*IBEA* [3]
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

components based 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.

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: maximum design and code reuse, flexibility and adaptability, utility, transparent and easy access to performance and robustness, portability, as well as usability and efficiency. These goals are presented in details in [20]. The ParadisEO platform honors all the above-mentioned criteria and aims to be used by both non-specialists and optimization experts.

B. ParadisEO and ParadisEO-MOEO

ParadisEO[1] 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)[2] [22], 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. Second, ParadisEO-MO contains a set of tools for single-solution based metaheuristics. Next, ParadisEO-MOEO is specifically dedicated to the reusable design of metaheuristics for multi-objective 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 multi-objective 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 in order to provide a modular architecture allowing to quickly and conveniently develop any new scheme with a minimum code writing. Moreover, 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 advances of the EMO field.

C. Implementation

This section gives a detailed description of the base classes provided within the ParadisEO framework to design an EMO algorithm[3]. The flexibility of the framework and its modular architecture based on the three main multi-objective 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. Due to space limitation, only UML diagrams of EMO-related components are given, but the whole inheritance diagram as well as classes documentation and examples of use are available on the ParadisEO website.

1) Representation: 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. 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-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

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

[2] `http://eodev.sourceforge.net`

[3] The classes presented in this paper are defined as in version 1.2 of ParadisEO.

for vectors composed of bits, of integers or of real-coded values that can thus directly be used in a ParadisEO-MOEO-designed application.

2) Initialization: A number of initialization schemes already exist 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.

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.

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. Moreover, variation mechanisms for some classical (real-coded, vector-based or permutation-based) representation are already provided in the framework. Note that all variation operators designed for a given problem must be embedded into a `eoTranform` object.

5) Fitness Assignment: Following the taxonomy introduced in Section III, the fitness assignment schemes are classified into four main categories, as illustrated in the UML diagram of Fig. 2: 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* [12], the *dominance-rank*, *dominance-count* and *dominance-depth* schemes, as well as the *indicator-based* fitness assignment strategy proposed in [3]. Moreover, a dummy fitness assignment strategy has been added in case it would be useful for some specific implementation.

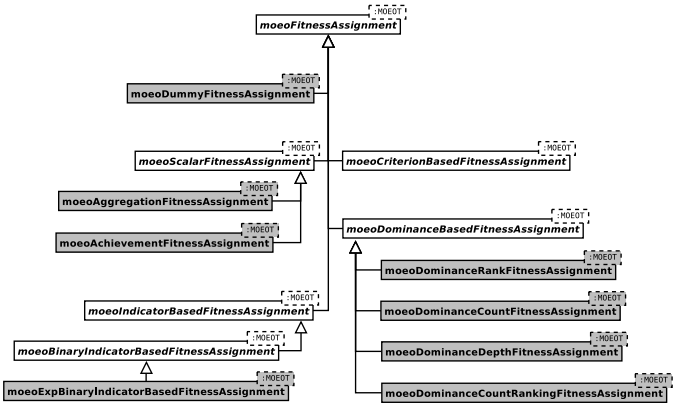

Fig. 2. UML diagram for fitness assignment

6) Diversity Assignment: As illustrated in Fig. 3, the diversity preservation strategy must inherit of the `moeoDiversityAssignment` class. A number diversity assignment schemes are already available, including sharing [17], crowding [18] and a nearest neighbor scheme [2].

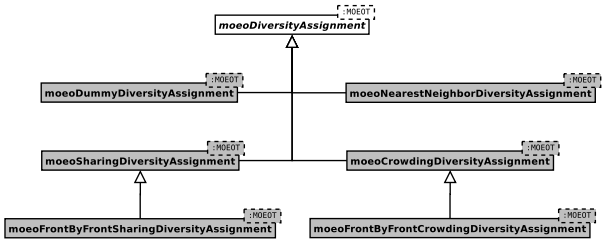

Fig. 3. UML diagram for diversity assignment

7) Selection: There exists a large number of selection strategies in the frame of EMO. Four ones are already provided within ParadisEO-MOEO. First, the *random selection* consists of selecting a parent randomly among the population members. 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 $(1-p)$. All these selection methods are of the `moeoSelectOne` type and needs to be embedded into a `eoSelect` object to be properly used.

8) Replacement: Three replacement schemes are provided within ParadisEO-MOEO. First, the *generational replacement* consists of keeping the offspring population only, while all parents are deleted. Next, the *elitist replacement* consists of choosing the N best solutions (where N stands for the population size). At last, the *environmental replacement*, that consists of repeatedly deleting the worst individual, and in updating the fitness and the diversity values of the remaining solutions each time there is a deletion.

9) Elitism: As shown in Fig. 4, 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, ParadisEO-MOEO offers the opportunity to use different dominance relation to update an archive contents including Pareto-dominance, weak-dominance, strict-dominance, ϵ-dominance [19], and g-dominance [15].

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

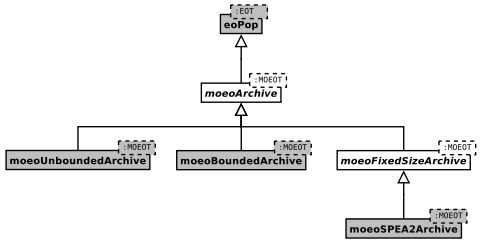

Fig. 4. UML diagram for archiving

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 useful 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 non-dominated front can be observed or study using graphical tools such as GUIMOO[4]. Furthermore, an important issue in the EMO field relates to the algorithm performance analysis and to set quality metrics [23]. A couple of metrics are featured within ParadisEO-MOEO, including the hypervolume metric in both its unary [10] and its binary [23] form, the additive and the multiplicative ϵ-indicators [23], etc. Another interesting feature is the possibility to compare the current archive with the archive of the previous iteration by using a given binary metric, and to print the progression of this measure iteration after iteration.

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 minimum effort. Fig. 5 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 (Multi-Objective 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 Section IV-C.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.

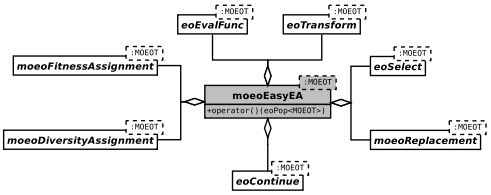

Fig. 5. A possible instantiation for the design of an EMO algorithm

In order to satisfy both the common user and the more experimented one, ParadisEO-MOEO also provides even more easy-to-use EAs. These classes propose different implementations of some state-of-the-art EMO algorithms by using the fine-grained components of ParadisEO. They are based on a simple combination of components, as described in Section III-B. Hence, MOGA [8], NSGA [9], NSGA-II [1], SPEA2 [2] and IBEA [3] are proposed in a way that a minimum number of problem- or algorithm-specific parameters are required. 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.

D. 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 [24]), scheduling problems (permutation flow-shop scheduling problem [25]), routing problems (multi-objective traveling salesman problem, bi-objective ring star problem [26]), etc. Moreover, it has been successfully employed to solve real-world applications in structural biology (docking problem [27]), feature selection in cancer classification [28], materials design in chemistry [29], etc. Besides, a detailed documentation as well as some tutorial lessons and problem-specific implementations are freely available on the ParadisEO website[5]. 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. 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 [26], a multi-objective cooperative island model has been designed in [30], and costly evaluation functions have been parallelized in [27]. Refer to [20] for more information about ParadisEO hybrid and parallel models.

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

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

V. Conclusion and Perspectives

In this paper, we gave a unified view of evolutionary algorithms to solve multi-objective optimization problems. We identified the common concepts for all evolutionary multi-objective optimization methods, and presented in details the main issues devoted to the multi-objective case: fitness assignment, diversity maintaining and elitism. As a result, we proposed a general purpose model based on a fine-grained decomposition, and we validated it by regarding a number of state-of-the-art methods as simple instances of the model. Then, we used this unified view as a starting point for the implementation of a software framework dedicated to the flexible and reusable design of evolutionary multi-objective metaheuristics, namely ParadisEO-MOEO. The main component taking part into the library for the design of a multi-objective evolutionary algorithm have been presented in details.

In the future, we will try to generalize the model proposed in this paper to other kinds of metaheuristics for multi-objective optimization, as we believe that a lot of identified components are shared by many MOP search methods. In particular, we will first investigate the common concepts involved in multi-objective local search or scatter search methods. Afterward, the resulting mechanisms will be implemented in a modular way and will be integrated into the ParadisEO-MOEO software framework.

References

[1] 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.

[2] 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.

[3] 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.

[4] 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, vol. 535. Springer-Verlag, 2004, pp. 3–38.

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

[6] 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.

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

[8] 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)*. Urbana-Champaign, IL, USA: Morgan Kaufmann, 1993, pp. 416–423.

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

[10] 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.

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

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

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

[14] 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.

[15] 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).

[16] 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.

[17] 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.

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

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

[20] 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.

[21] 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, vol. 4403. Matsushima, Japan: Springer-Verlag, 2007, pp. 386–400.

[22] 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 2001)*, Le Creusot, France, 2001, pp. 231–244.

[23] 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.

[24] 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*. Springer, 2005, ch. 6, pp. 105–145.

[25] 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, vol. 4403. Matsushima, Japan: Springer-Verlag, 2007, pp. 457–471.

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] 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.