View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by INRIA a CCSD electronic archive server

A software framework based on a conceptual unified model for evolutionary multiobjective optimization: ParadisEO-MOEO

Arnaud Liefooghe, Laetitia Jourdan, El-Ghazali Talbi

To cite this version:

Arnaud Liefooghe, Laetitia Jourdan, El-Ghazali Talbi. A software framework based on a conceptual unified model for evolutionary multiobjective optimization: ParadisEO-MOEO. European Journal of Operational Research, Elsevier, 2011, 209 (2), pp.104-112. 10.1016/j.ejor.2010.07.023 . hal-00522612

HAL Id: hal-00522612

https://hal.archives-ouvertes.fr/hal-00522612

Submitted on 1 Oct 2010

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

European Journal of Operational Research xxx (2010) xxx–xxx

Contents lists available at ScienceDirect

European Journal of Operational Research

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

Decision Support

A software framework based on a conceptual unified model for evolutionary multiobjective optimization: ParadisEO-MOEO

Arnaud Liefooghe [a,b,*], Laetitia Jourdan [a], El-Ghazali Talbi [a,c]

[a] Université Lille 1, Laboratoire d'Informatique Fondamentale de Lille, UMR CNRS 8022, INRIA Lille-Nord Europe, Parc Scientifique de la Haute Borne, 40 Avenue Halley, 59650 Villeneuve d'Ascq, France
[b] University of Coimbra, CISUC, Department of Informatics Engineering, Portugal
[c] King Saud University, Riyadh, Saudi Arabia

ARTICLE INFO

Article history:
Received 24 April 2009
Accepted 24 July 2010
Available online xxxx

Keywords:
Multiple objective programming
Evolutionary algorithms
Conceptual unified model
Algorithm design and implementation
Software framework

ABSTRACT

This paper presents a general-purpose software framework dedicated to the design and the implementation of evolutionary multiobjective optimization techniques: ParadisEO-MOEO. A concise overview of evolutionary algorithms for multiobjective optimization is given. A substantial number of methods has been proposed so far, and an attempt of conceptually unifying existing approaches is presented here. Based on a fine-grained decomposition and following the main issues of fitness assignment, diversity preservation and elitism, a conceptual model is proposed and is validated by regarding a number of state-of-the-art algorithms as simple variants of the same structure. This model is then incorporated into the ParadisEO-MOEO software framework. This framework has proven its validity and high flexibility by enabling the resolution of many academic, real-world and hard multiobjective optimization problems.
© 2010 Elsevier B.V. All rights reserved.

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 $n \geqslant 2$ objective functions $(f_1, f_2, \ldots, f_n)$, and a set X of feasible solutions in the *decision space*. Let Z be the set 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 $f : X \to Z$ 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 $\forall i \in \{1,2,\ldots,n\}$, $z_i \leqslant z'_i$ 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 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 *non-dominated* (or *efficient*, *Pareto optimal*) if $f(x)$ maps to a non-dominated point. The set of all efficient solutions is called the *efficient set* (or *Pareto optimal set*) and its mapping in the objective space is the *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 region of interest 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 (Deb, 2001; Coello Coello et al., 2007).

As pointed out by different authors (see, *e.g.*, Coello Coello et al., 2007; Zitzler et al., 2004), approximating an efficient set is itself a

* Corresponding author at: Université Lille 1, Laboratoire d'Informatique Fondamentale de Lille, UMR CNRS 8022, INRIA Lille-Nord Europe, Parc Scientifique de la Haute Borne, 40 avenue Halley, 59650 Villeneuve d'Ascq, France. Tel.: +33 3 59 57 79 30; fax: +33 3 59 57 78 50.
E-mail addresses: arnaud.liefooghe@lifl.fr (A. Liefooghe), laetitia.jourdan@lifl.fr (L. Jourdan), talbi@lifl.fr (E.-G. Talbi).

0377-2217/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.ejor.2010.07.023

Please cite this article in press as: Liefooghe, A., et al. A software framework based on a conceptual unified model for evolutionary multiobjective optimization: ParadisEO-MOEO. European Journal of Operational Research (2010), doi:10.1016/j.ejor.2010.07.023

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 the existing approaches, one may cite VEGA (Schaffer, 1985), MOGA (Fonseca and Fleming, 1993), NSGA (Srinivas and Deb, 1994), NSGA-II (Deb et al., 2002), NPGA (Horn et al., 1994), SPEA (Zitzler and Thiele, 1999), SPEA2 (Zitzler et al., 2001) or PESA (Corne et al., 2000). 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 Zitzler et al. (2003) for a review.

Zitzler et al. (2004) notice that initial EMO approaches were mainly focused on moving toward the Pareto front (Schaffer, 1985; Fourman, 1985). Afterwards, diversity preservation mechanisms quickly emerged (Fonseca and Fleming, 1993; Srinivas and Deb, 1994; Horn et al., 1994). Then, at the end of the 90s, the concept of elitism, related to the preservation and use of non-dominated solutions, became very popular and is now employed in most recent EMO methods (Zitzler and Thiele, 1999; Zitzler et al., 2001; Knowles and Corne, 2000). 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 (Coello Coello et al., 2007; Zitzler et al., 2004). Based on these three main notions, several attempts have been made in the past for unifying EMO algorithms. Laumanns (2000) focus on elitist EMO search methods. Their study has been later extended by Zitzler et al. (2004) where the algorithmic concepts of fitness assignment, diversity preservation and elitism are largely discussed. More recently, Deb (2008) proposed a robust framework for EMO based on NSGA-II (Non-dominated Sorting Genetic Algorithm, Deb et al., 2002). The latter approach is decomposed into three main issues related to elite preservation, non-dominated 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 ingredients are shared by many EMO algorithms, so that, somehow, they can all be seen as variants of the same conceptual model, as it will be highlighted in the remainder of the paper. Furthermore, some of these existing conceptual models have been used as a basis for the design of tools to help practitioners and researchers for MOP solving. For instance, following Laumanns (2000) and Zitzler et al. (2004), the authors proposed a software framework for EMO called PISA (Bleuler et al., 2003). 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 until then, and will be discussed later in the paper.

The purpose of the present work is to present a conceptual framework and its practical integration into the ParadisEO-MOEO software. Firstly, a conceptual model for EMO is given and will serve as a basis and motivation for the design of the software framework. We describe the basic issues shared by many algorithms, and we introduce a general-purpose model as well as a classification of its fine-grained elements. 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 (Deb et al., 2002), SPEA2 (Zitzler et al., 2001) and IBEA (Zitzler and Künzli, 2004) are taken as examples. Finally, 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, which is available for download at the following URL: http://paradiseo.gforge.inria.fr. 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 conceptual 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 Section 2, a concise, unified and up-to-date presentation of EMO techniques are discussed. A conceptual model is introduced as a basis for the design of our software framework. Next, a motivated presentation of ParadisEO-MOEO is given in Section 3, and is followed by a short description of the design and the implementation of EMO algorithms under ParadisEO-MOEO. Finally, the last section concludes the paper.

2. A conceptual unified model for evolutionary multiobjective optimization

An evolutionary algorithm (EA) (Eiben and Smith, 2003) is a search method that belongs to the class of metaheuristics (Talbi, 2009), 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. This search process is iterated until a given stopping condition 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 approximation to be found also has to be well-spread over the objective space, so that a diversity preservation mechanism is usually required. This fitness and diversity information are required 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.

Please cite this article in press as: Liefooghe, A., et al. A software framework based on a conceptual unified model for evolutionary multiobjective optimization: ParadisEO-MOEO. European Journal of Operational Research (2010), doi:10.1016/j.ejor.2010.07.023

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

2.1. Elements description

This section provides a description of the elements involved in our conceptual unified model. EMO-related issues 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. An 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 require a proper solution representation. Various encodings may be used such as binary variables, real-coded vectors, permutations, discrete vectors, and more complex representations. 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 set, *i.e.*, the value corresponding to each objective function must be evaluated.

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 of a 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 the works of Zitzler et al. (2004) and Coello Coello et al. (2007), 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 (see Miettinen, 1999).
- *Criterion-based approaches*, where each objective function is treated separately. For instance, in VEGA (vector evaluated genetic algorithm) (Schaffer, 1985), 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 (Fourman, 1985), a hierarchical order is defined beforehand between objective functions.
- *Dominance-based approaches*, where a dominance relation is used to classify solutions. For instance, *dominance-rank* techniques compute the number of population items that dominate a given solution (Fonseca and Fleming, 1993). Such a strategy takes part in, *e.g.*, Fonseca and Fleming MOGA (multiobjective genetic algorithm) (Fonseca and Fleming, 1993). In *dominance-count* techniques, the fitness value of a solution corresponds to the number of individuals that are dominated by these solutions (Zitzler and Thiele, 1999). Finally, *dominance-depth* strategies consist of classifying a set of solutions into different classes (or fronts) (Goldberg, 1989). 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 non-dominated set, individuals from the second front all belong to the second best non-dominated set, and so on. The latter approach is used in both NSGA (non-dominated sorting genetic algorithm) (Srinivas and Deb, 1994) and NSGA-II (Deb et al., 2002). In addition, note that several schemes can also be combined, as in the case of, for example, Zitzler and Thiele (1999). In the frame of dominance-based approaches, the most commonly used dominance relation is based on Pareto-dominance as given in Section 1. But some recent techniques are based on other dominance operators such as ϵ-dominance in Deb et al. (2005a) and g-dominance in Molina et al. (2009).
- *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) proposed by Zitzler and Künzli (2004) or SMS-EMOA (S-metric selection EMO algorithm) of Beume et al. (2007).

2.1.6. Diversity assignment

As noticed in the previous section, aiming at approximating the efficient set is not only an issue related to 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 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. Nevertheless, a distance metric partly or fully defined in the parameter space strongly relies on the tackled problem.

Please cite this article in press as: Liefooghe, A., et al. A software framework based on a conceptual unified model for evolutionary multiobjective optimization: ParadisEO-MOEO. European Journal of Operational Research (2010), doi:10.1016/j.ejor.2010.07.023

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 (1987) to preserve diversity among the solutions of an EA population. It has first been employed by Fonseca and Fleming (1993) 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*. Another diversity assignment scheme is the concept of *crowding*, first suggested by Holland (1975) and used by De Jong (1975) to prevent *genetic drift*. It is employed by Deb et al. (2002) 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 an individual, the higher 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 non-dominated solutions also contribute to the evolution engine. Such an approach has successfully been applied in various elitist EMO algorithms including SPEA (Zitzler and Thiele, 1999), SPEA2 (Zitzler et al., 2001) and PESA (Corne et al., 2000). In addition, in order to prohibit the crossover of dissimilar parents, mating restriction (Goldberg, 1989) 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 populations. In generational replacement, the offspring population systematically replaces the parent one. An elitist strategy consists of preserving the N best solutions from both populations (N stands for the appropriate population size), either by means of a one-shot deletion or by deleting the worst solution iteratively until the required population size is reached, so that fitness and diversity information of remaining solutions can be updated each time there is a deletion.

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 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 relation. But, in the literature, other dominance criterion can be found. Examples are weak-dominance, strict-dominance, ϵ-dominance (Helbig and Pateva, 1994; Laumanns et al., 2002), and so on. When dealing about archiving, one may distinguishe 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 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(s). 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 within SPEA2 (Zitzler et al., 2001). 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, as proposed by Zitzler and Thiele (1999), see Section 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 algorithms as instances of the conceptual unified model

By means of the conceptual model presented 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 elements presented above. In Table 1, three EMO approaches, namely NSGA-II (Deb et al., 2002), SPEA2 (Zitzler et al., 2001) and IBEA (Zitzler and Künzli, 2004), are regarded as simple instances of the model proposed in this paper. For obvious reasons, only problem-independent issues 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 model for EMO, what strongly validates the proposed approach. But other examples can be found in the literature. For instance, the only part that differs from NSGA (Srinivas and Deb, 1994) to NSGA-II 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 by Deb et al. (2005a). This algorithm is a

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

Elements	NSGA-II	SPEA2	IBEA
Fitness assignment	Dominance-depth	Dominance-count and rank	Binary quality indicator
Diversity assignment	Crowding distance	kth nearest neighbor	None
Selection	Binary tournament	Elitist binary tournament	Binary tournament
Replacement	One-shot elitist replacement	Generational replacement	Iterative elitist replacement
Archiving	None	Fixed-size archive	None
Stopping criteria	Number of generations	Number of generations	Number of generations

Please cite this article in press as: Liefooghe, A., et al. A software framework based on a conceptual unified model for evolutionary multiobjective optimization: ParadisEO-MOEO. European Journal of Operational Research (2010), doi:10.1016/j.ejor.2010.07.023

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 by Molina et al. (2009) is experimented by the authors on an 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 of evolutionary multiobjective optimization algorithms 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 Ph.D. work of Sébastien Cahon, supervised by Nouredine Melab and El-Ghazali Talbi (Cahon et al., 2004). 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 Liefooghe et al. (2007b). 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 conceptual model presented above.

3.1. Motivations

In practice, there exists a large diversity of optimization problems to be solved, engendering wide possibilities in terms of 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 design new algorithms, to integrate new elements into an existing method, or even to combine different search mechanisms. Moreover, such a tool is of large interest in order to be able to evaluate and to compare different algorithms fairly.

Hence, as pointed out in Cahon et al. (2004) and Talbi (2009), three major approaches exist for the development of metaheuristics: *from scratch* or *no reuse*, *code reuse only* and *both design and code reuse*. Firstly, programmers are tempted to develop and implement their own code from scratch. However, it requires time and energy and the resulting code is generally error prone and difficult to maintain and evolve. The second approach consists of reusing a third-party source code available on the web, either as individual programs or as libraries. Individual programs often have application-dependent sections that are to be extracted before a new application-dependent code is to be inserted. Similarly, modifying these sections is often time consuming and error prone. Code reuse through libraries is obviously better because they are often well tried, tested, documented, and thus more reliable. However, libraries do not allow the reuse of the complete invariant part of the algorithms related to the design. Therefore, the code effort remains important. At last, both design and code reuse allow to overcome this problem. 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 building blocks based on a strong conceptual separation of the invariant part and the problem-specific part of metaheuristics. Thus, each time a new optimization problem is to be tackled, both code and design can directly be reused in order to redo as little code as possible. Hence, the implementation effort is minimal with regards to the problem under investigation. Generally speaking, the constant part is encapsulated in generic or abstract skeletons that are implemented in the framework. The variable part, which is problem specific, is fixed in the framework but must be supplied by the user. These user-defined functions are thus to be called by the framework. To do so, the design of the framework must be based on a clear conceptual separation between the resolution methods and the problem to be solved. Object-oriented design and programming are generally recommended for such a purpose. But another way to perform this separation is to provide a set of modules for each part, and to make them cooperate thought text files. However, this allows less flexibility than the object-oriented approach, and the execution is generally much more time-consuming. Besides, note that two types of software frameworks can be distinguished: white-box and black-box frameworks.

3.2. ParadisEO and ParadisEO-MOEO

ParadisEO (http://paradiseo.gforge.inria.fr) is a white-box object-oriented software framework dedicated to the flexible design of metaheuristics for optimization problems of continuous, discrete and combinatorial nature. Based on EO (evolving objects, http://eodev.sourceforge.net) (Keijzer et al., 2001), this template-based C++ computation library is portable across both Unix-like and Windows systems. This software is governed by the CeCILL license under French law and abiding by the rules of distribution of free software (http://www.cecill.info). ParadisEO tends to be used by both non-specialists and optimization experts. As illustrated in Fig. 1, it 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 in-

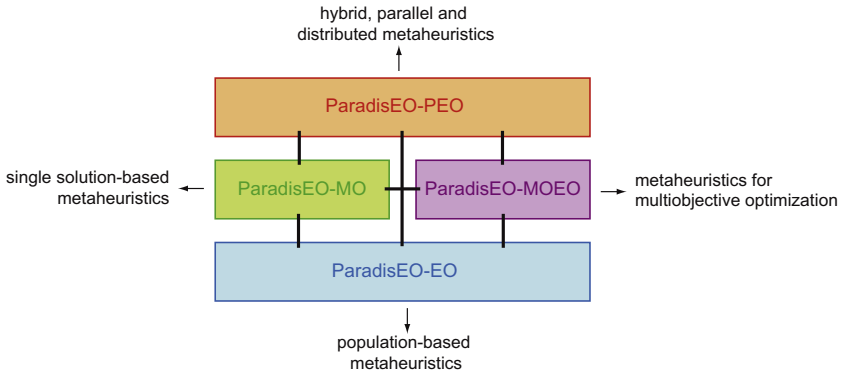

Fig. 1. The modules constituting the ParadisEO framework.

Please cite this article in press as: Liefooghe, A., et al. A software framework based on a conceptual unified model for evolutionary multiobjective optimization: ParadisEO-MOEO. European Journal of Operational Research (2010), doi:10.1016/j.ejor.2010.07.023

tended to solve. This separation confers a maximum code and design reuse to the user. The first module, ParadisEO-EO, provides a broad range of classes for the development of population-based metaheuristics, including evolutionary algorithms and 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, which is of our interest in this paper, 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 model presented in the previous section and is conceptually divided into fine-grained elements. 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 elements 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 algorithmic part required for their implementation. ParadisEO-MOEO constantly evolves and new features might be regularly added to the framework in order to provide 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 (Cahon et al., 2004; Talbi, 2009):

- *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, so that it 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 algorithmic elements. Users must have access to source code and use inheritance or specialization concepts of object-oriented programming to derive new objects from base or abstract classes. Furthermore, as existing problems evolve and new others arise, the framework must be conveniently specialized and adapted.
- *Utility*. The framework must cover a broad range of metaheuristics, problems, parallel and distributed models, hybridization mechanisms, etc. Of course, advanced features must not add any difficulty for users wanting to implement classical algorithms.
- *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).
- *Easy-of-use 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 intended 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 a few particular EMO strategies, such as ECJ (http://cs.gmu.edu/eclab/projects/ecj/), JavaEVA (Streichert and Ulmer, 2005) and Open BEAGLE (Gagné and Parizeau, 2006). Table 2 gives a non-exhaustive comparison between a number of existing software frameworks for EMO, including jMetal (Durillo et al., 2006), the MOEA toolbox for Matlab (Tan et al., 2001), MOMHLib++ (http://home.gna.org/momh/), PISA (Bleuler et al., 2003) and Shark (Igel et al., 2008). Note that other software packages exist for multiobjective optimization (Poles et al., 2008), 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),

Table 2
Main characteristics of 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

Please cite this article in press as: Liefooghe, A., et al. A software framework based on a conceptual unified model for evolutionary multiobjective optimization: ParadisEO-MOEO. European Journal of Operational Research (2010), doi:10.1016/j.ejor.2010.07.023

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 mention 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 objects. 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 building blocks 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 by Talbi (2002), 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

Technical details on the implementation of EMO algorithms under ParadisEO-MOEO can be found on the ParadisEO website (http://paradiseo.gforge.inria.fr), together with a complete documentation and many examples of use. The high flexibility of the framework and its modular architecture based on the three main multiobjective metaheuristic design issues (fitness assignment, diversity preservation and elitism) allow to implement efficient algorithms in solving a large diversity of MOPs. The granular decomposition of ParadisEO-MOEO is based on the conceptual model introduced in the previous section.

3.5.1. EMO algorithms

An EMO algorithm can easily be designed using the fine-grained building blocks of ParadisEO. Different operators can be experimented without engendering significant modifications in terms of code writing. A wide range of strategies are already provided, but this list is not exhaustive as the framework perpetually evolves and offers all that is necessary to develop new ones with a minimum effort. Indeed, ParadisEO is a white-box framework that tends to be flexible while being as user-friendly as possible.

In order to satisfy both the beginners and the more experimented users, ParadisEO-MOEO also provides even more easy-to-use EMO algorithms. These classes propose different implementations of some state-of-the-art methods. They are based on a simple combination of building blocks, as described in Section 2.2. Hence, MOGA (Fonseca and Fleming, 1993), NSGA (Srinivas and Deb, 1994), NSGA-II (Deb et al., 2002), SPEA2 (Zitzler et al., 2001), IBEA (Zitzler and Künzli, 2004) and SEEA (Liefooghe et al., 2010) 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 function is the single part to be implemented. These easy-to-use algorithms also tend 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.

3.5.2. Parallel EMO algorithms

The design and the implementation of sequential EMO algorithms is just an aspect of the features provided by ParadisEO. Indeed, EMO algorithms can easily be hybridized, parallelized and distributed, thanks to the PEO module of ParadisEO (Cahon et al., 2004). The framework has been designed in such a way that the user can, for instance, parallelize his/her sequential EMO algorithm very easily and transparently. When solving real-world optimization problems, running simple EMO algorithms on a large population or on complex individuals often requires high computational resources. However, since each population member can be seen as an independent unit, parallelism naturally arises while dealing with population-based metaheuristics. Melab et al. (2006) identify three levels of parallelism for evolutionary computation, and then EMO algorithms: the solution level, the iteration level and the algorithmic level. ParadisEO is one of the rare framework that provides these most common parallel and distributed models. The resulting applications are portable on different execution platforms such as parallel computing, cluster computing, Internet computing and grid computing.

Evaluating a solution in the objective space is by far the most computationally expensive step of any metaheuristic for difficult or large-size optimization problems. Firstly, at the solution level, the evaluation of a single solution can be parallelized, by either partitioning data or objective functions. However, such a parallel evaluation function strongly relies on the problem to be solved so that the user must have enough knowledge about parallel computing to implement it efficiently. Second, at the iteration level, the evaluation of the evolving population can be parallelized as well. At each algorithm iteration, the offspring solutions are distributed to different workers in order to be evaluated in parallel. In terms of implementation, the latter strategy can be done by embedding the original, either sequential or parallel, evaluation function into a more advanced object with which the EMO algorithm is to be created. Note that these two first parallelization schemes do not modify the behavior of the EMO algorithm but are essentially focused on speeding up the search. At last, the algorithmic level consists of running simultaneously a number of EMO algorithms in parallel. This can be done in a multi-start way, what seems to be particularly interesting for testing different independent algorithms or parameter settings. However, various EMO algorithms can also cooperate with each other during the search process, either in a centralized way or like in the island model. The approach results in a hybrid co-evolutionary EMO algorithm where different agents cooperate in a high-level teamwork mode (Talbi, 2002). For instance, in the island model, sub-populations are distributed in a set of islands in which semi-isolated EMO algorithms are executed in parallel. Each island manages a proper population and an optional proper archive. However, a global archive containing the set of non-dominated solutions from all the islands can possibly be designed as well. The migration of individuals from an island to another is managed by a migration decision condition, an exchange topology, a number of emigrants, an emigrants selection scheme and an integration strategy. Note that emigrant individuals can be selected from both the current population and the current archive of a given island. The reader is referred to (Cahon et al., 2004) for more information about the ParadisEO parallel models.

3.6. Discussion

We believe that the aforementioned characteristics make ParadisEO-MOEO a valuable tool for both researchers and practitioners, and a unique software framework in comparison to existing ones. Indeed, it includes many state-of-the-art and up-to-date EMO algorithms. The rich set of ParadisEO modular ingredients

Please cite this article in press as: Liefooghe, A., et al. A software framework based on a conceptual unified model for evolutionary multiobjective optimization: ParadisEO-MOEO. European Journal of Operational Research (2010), doi:10.1016/j.ejor.2010.07.023

serve as building blocks to implement these methods. The related source code is maintained and regularly updated by the developers. Moreover, the framework gives the possibility to design and implement a wide number of new MOP resolution methods, either sequential or parallel, just by combining existing elements in an innovative way, or by implementing original ones. As an example, to illustrate the high ParadisEO flexibility, the dominance relation used to manage the archive of a given EMO algorithm is just a parameter of the corresponding object. The latter can then be modified from, say, Pareto-dominance to ϵ-dominance with insignificant effort in order to obtain a different method. Furthermore, analogously to the design level, a large number of issues are shared by many EMO algorithms at the implementation level as well. Hence, ParadisEO can serve as a reference implementation in order to compare different algorithms fairly. For instance, whenever a new EMO algorithm is proposed, its efficiency can be experimentally demonstrated by comparing its behavior with existing ones.

On the other hand, ParadisEO is also a practical tool that can be used to tackle an original MOP. The implementation of efficient programs is highly facilitated so that the user only has to focus on problem-related issues of representation, initialization, evaluation and variation. The implementation effort is even more reduced when a classical solution representation can be applied for the problem under consideration. By means of classical representation, we include real-coded, binary, integer, permutation variable representations, and so on. For such problems, the development and time cost are reduced to minimum since the evaluation function is the single element to be implemented. Of course, this cost is always related to the proficiency of the programmer in charge of the implementation. Once this evaluation function is available, the user only has to instantiate any EMO algorithm (NSGA-II, SPEA2, IBEA, ...) and optionally any parallel model (parallel evaluation, island model, ...) to obtain a powerful resolution program that is able to run on a large range of material architectures (sequential, parallel, distributed) and their associated operating systems (Windows, Linux, MacOS). Therefore, the remaining development consists of a direct instantiation of selected strategies. Though, for more sophisticated solution encodings, the development cost remains substantial with respect to the complexity of the underlying representation and to the level of expertise of the programmer. But it will always be lower than implementing a whole specific EMO algorithm from scratch. Otherwise, starting from a single-objective optimization problem implemented within ParadisEO, it is a commonplace to investigate a multiobjective variant, so that existing EMO algorithms can then be instantiated to solve the resulting MOP.

With regards to runtime efficiency, it is difficult, if not impracticable, to show how ParadisEO-MOEO performs in comparison to existing software frameworks or even to a specific implementation done from scratch. Indeed, it will always depend on the programmer ability to handle such a tool, and may often depend on the problem to be solved and the algorithm to be applied. Development from scratch generally allows a deep code optimization with respect to a very specific application, but this requires a high level of expertise and is also time-consuming in terms of development cost. Additionally, ParadisEO-MOEO has a clear advantage of not communicating information through text files, as done in PISA for instance, what is known to be very expensive in time.

Finally, ParadisEO-MOEO has been used and experimented to solve a large range of MOPs from both academic and real-world fields, which validates its high flexibility. Indeed, various academic MOPs have been tackled, including continuous test functions such as the ZDT and DTLZ functions family (Deb et al., 2005b) or the Schaffer problem (Schaffer, 1985), scheduling problems including permutation flow-shop scheduling (Liefooghe et al., 2007a), routing problems such as the multiobjective traveling salesman problem and the bi-objective ring star problem (Liefooghe et al., 2010). Moreover, ParadisEO-MOEO has been successfully applied to solve real-world applications in structural biology (Boisson et al., 2008), feature selection in cancer classification (Talbi et al., 2008), materials design in chemistry (Schuetze et al., 2008), portfolio optimization, etc. Note that the problem-related part of some aforementioned applications is freely available as *contributions* on the ParadisEO website (http://paradiseo.gforge.inria.fr), together with a detailed documentation and some tutorial lessons.

Additionally, hybrid and parallel metaheuristics have also been designed within ParadisEO to solve MOPs. For instance, hybrid EMO algorithms have been experimented by Liefooghe et al. (2010), a multiobjective cooperative island model has been designed by Talbi et al. (2007), costly evaluation functions have been parallelized by Boisson et al. (2008), and a parallel multiple reference point approach has been proposed by Figueira et al. (2010).

4. Concluding remarks

The current paper described a software framework for the development of evolutionary multiobjective optimization algorithms. First, we identified the common concepts shared by many evolutionary multiobjective optimization techniques, separating the problem-specific part from the invariant part of such approaches. We emphasized the main issues of fitness assignment, diversity preservation and elitism. Therefore, we presented a conceptual model, based on a fine-grained decomposition, and we illustrated its robustness and reliability by treating a number of state-of-the-art algorithms as simple instances of it. Next, this unified view has been used as a starting point for the design and the implementation of a general-purpose software framework 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 issues from evolutionary multiobjective optimization are shared by many other metaheuristic methodologies. Thereafter, we plan to generalize the conceptual model introduced in this paper to other existing metaheuristic approaches for multiobjective optimization. Hence, multiobjective local search or scatter search methods might be interesting paths to explore in order to investigate their ability and their modularity for providing such a flexible model as the one presented in this paper. The resulting general-purpose models and their particular mechanisms would then be integrated into the ParadisEO-MOEO software framework.

Acknowledgments

This work was supported by the ANR DOCK project. The authors would like to gratefully acknowledge Jérémie Humeau, Thomas Legrand, 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. Moreover, the authors would like to thank the anonymous reviewers for their valuable comments and suggestions.

Please cite this article in press as: Liefooghe, A., et al. A software framework based on a conceptual unified model for evolutionary multiobjective optimization: ParadisEO-MOEO. European Journal of Operational Research (2010), doi:10.1016/j.ejor.2010.07.023

References

Beume, N., Naujoks, B., Emmerich, M., 2007. SMS-EMOA: Multiobjective selection based on dominated hypervolume. European Journal of Operational Research 181 (3), 1653–1669.

Bleuler, S., Laumanns, M., Thiele, L., Zitzler, E., 2003. PISA – A platform and programming language independent interface for search algorithms. In: Second International Conference on Evolutionary Multi-Criterion Optimization (EMO 2003). Lecture Notes in Computer Science, vol. 2632. Springer-Verlag, Faro, Portugal, pp. 494–508.

Boisson, J.-C., Jourdan, L., Talbi, E.-G., Horvath, D., 2008. 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, pp. 187–194.

Cahon, S., Melab, N., Talbi, E.-G., 2004. ParadisEO: A framework for the reusable design of parallel and distributed metaheuristics. Journal of Heuristics 10 (3), 357–380.

Coello Coello, C.A., Lamont, G.B., Van Veldhuizen, D.A., 2007. Evolutionary Algorithms for Solving Multi-Objective Problems, second ed. Springer, New York, USA.

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

Deb, K., 2001. Multi-Objective Optimization using Evolutionary Algorithms. John Wiley & Sons, Chichester, UK.

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

Deb, K., Agrawal, S., Pratap, A., Meyarivan, T., 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation 6 (2), 182–197.

Deb, K., Mohan, M., Mishra, S., 2005a. Evaluating the ϵ-domination based multi-objective evolutionary algorithm for a quick computation of pareto-optimal solutions. Evolutionary Computation 13 (4), 501–525.

Deb, K., Thiele, L., Laumanns, M., Zitzler, E., 2005b. Scalable test problems for evolutionary multi-objective optimization. In: Abraham, A., Jain, R., Goldberg, R. (Eds.), Evolutionary Multiobjective Optimization: Theoretical Advances and Applications. Springer, pp. 105–145 (Chapter 6).

De Jong, K.A., 1975. An Analysis of the Behavior of a Class of Genetic Adaptive Systems. Ph.D. Thesis, Ann Arbor, University of Michigan.

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

Eiben, A.E., Smith, J.E., 2003. Introduction to Evolutionary Computing. Springer, New York, USA.

Figueira, J., Liefooghe, A., Talbi, E.-G., Wierzbicki, A., 2010. A parallel multiple reference point approach for multi-objective optimization. European Journal of Operational Research 205, 390–400.

Fonseca, C.M., Fleming, P.J., 1993. Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. In: 5th International Conference on Genetic Algorithms (ICGA 1993). Morgan Kaufmann, Urbana-Champaign, IL, USA, pp. 416–423.

Fourman, M.P., 1985. Compaction of symbolic layout using genetic algorithms. In: 1st International Conference on Genetic Algorithms (ICGA 1985). Lawrence Erlbaum Associates, Pittsburgh, PA, USA, pp. 141–153.

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

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

Goldberg, D.E., Richardson, J., 1987. Genetic algorithms with sharing for multimodal function optimization. In: 2nd International Conference on Genetic Algorithms and their application. Lawrence Erlbaum Associates, Inc., Mahwah, NJ, USA, pp. 41–49.

Helbig, S., Pateva, D., 1994. On several concepts for ϵ-efficiency. OR Spektrum 16 (3), 179–186.

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

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

Igel, C., Glasmachers, T., Heidrich-Meisner, V., 2008. Shark. Journal of Machine Learning Research 9, 993–996.

Keijzer, M., Merelo, J.-J., Romero, G., Schoenauer, M., 2001. Evolving objects: A general purpose evolutionary computation library. In: 5th International Conference on Artificial Evolution (EA 2001). Le Creusot, France, pp. 231–244.

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

Laumanns, M., Zitzler, E., Thiele, L., 2000. A unified model for multi-objective evolutionary algorithms with elitism. In: IEEE Congress on Evolutionary Computation (CEC 2000). IEEE Press, Piscataway, New Jersey, USA, pp. 46–53.

Laumanns, M., Thiele, L., Deb, K., Zitzler, E., 2002. Combining convergence and diversity in evolutionary multi-objective optimization. Evolutionary Computation 10 (3), 263–282.

Liefooghe, A., Basseur, M., Jourdan, L., Talbi, E.-G., 2007a. Combinatorial optimization of stochastic multi-objective problems: an application to the flow-shop scheduling problem. In: Fourth International Conference on Evolutionary Multi-Criterion Optimization (EMO 2007). Lecture Notes in Computer Science, vol. 4403. Springer-Verlag, Matsushima, Japan, pp. 457–471.

Liefooghe, A., Basseur, M., Jourdan, L., Talbi, E.-G., 2007b. ParadisEO-MOEO: A framework for evolutionary multi-objective optimization. In: Fourth International Conference on Evolutionary Multi-Criterion Optimization (EMO 2007). Lecture Notes in Computer Science, vol. 4403. Springer-Verlag, Matsushima, Japan, pp. 386–400.

Liefooghe, A., Jourdan, L., Talbi, E.-G., 2010. Metaheuristics and cooperative approaches for the bi-objective ring star problem. Computers & Operations Research 37 (6), 1033–1044.

Melab, N., Talbi, E.-G., Cahon, S., Alba, E., Luque, G., 2006. Parallel metaheuristics: Models and frameworks. In: Talbi, E.-G. (Ed.), Parallel Combinatorial Optimization. John Wiley & Sons, Chichester, UK, pp. 149–162 (Chapter 6).

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

Molina, J., Santana, L.V., Hernández-Díaz, A.G., Coello Coello, C.A., Caballero, R., 2009. g-dominance: Reference point based dominance for multiobjective metaheuristics. European Journal of Operational Research 167 (2), 685–692.

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

Schaffer, J.D., 1985. Multiple objective optimization with vector evaluated genetic algorithms. In: 1st International Conference on Genetic Algorithms (ICGA 1985). Lawrence Erlbaum Associates, Pittsburgh, PA, USA, pp. 93–100.

Schuetze, O., Jourdan, L., Legrand, T., Talbi, E.-G., Wojkiewicz, J.-L., 2008. New analysis of the optimization of electromagnetic shielding properties using conducting polymers and a multi-objective approach. Polymers for Advanced Technologies 19 (7), 762–769.

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

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

Talbi, E.-G., 2002. A taxonomy of hybrid metaheuristics. Journal of Heuristics 8 (2), 541–564.

Talbi, E.-G., 2009. Metaheuristics: From Design to Implementation. Wiley.

Talbi, E.-G., Cahon, S., Melab, N., 2007. Designing cellular networks using a parallel hybrid metaheuristic on the computational grid. Computer Communications 30 (4), 698–713.

Talbi, E.-G., Jourdan, L., Garcia-Nieto, J., Alba, E., 2008. 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 Press, pp. 45–52.

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

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

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

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

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

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

Please cite this article in press as: Liefooghe, A., et al. A software framework based on a conceptual unified model for evolutionary multiobjective optimization: ParadisEO-MOEO. European Journal of Operational Research (2010), doi:10.1016/j.ejor.2010.07.023