ParadisEO-MOEO: A Framework for Evolutionary Multi-objective Optimization

Arnaud Liefooghe, Matthieu Basseur, Laetitia Jourdan, and El-Ghazali Talbi

INRIA Futurs, Laboratoire d'Informatique Fondamentale de Lille (LIFL), CNRS
Bât. M3, Cité Scientifique, 59655 Villeneuve d'Ascq cedex, France
{liefooga,basseur,jourdan,talbi}@lifl.fr

Abstract. This paper presents ParadisEO-MOEO, a white-box object-oriented generic framework dedicated to the flexible design of evolutionary multi-objective algorithms. This paradigm-free software embeds some features and techniques for Pareto-based resolution and aims to provide a set of classes allowing to ease and speed up the development of computationally efficient programs. It is based on a clear conceptual distinction between the solution methods and the multi-objective problems they are intended to solve. This separation confers a maximum design and code reuse. ParadisEO-MOEO provides a broad range of archive-related features (such as elitism or performance metrics) and the most common Pareto-based fitness assignment strategies (MOGA, NSGA, SPEA, IBEA and more). Furthermore, parallel and distributed models as well as hybridization mechanisms can be applied to an algorithm designed within ParadisEO-MOEO using the whole version of ParadisEO. In addition, GUIMOO, a platform-independant free software dedicated to results analysis for multi-objective problems, is briefly introduced.

Key words: object-oriented frameworks, design and code reuse, multi-objective optimization, evolutionary algorithms

1 Introduction

Nowadays, the usefulness of Multi-Objective Optimization (MOO) is globally established in the whole operational research community. Furthermore, evolutionary algorithms (EAs) are commonly used to solve multi-criterion problems since they naturally found a well-diversified set of good-quality solutions. EAs [12] are stochastic optimization processes based on an iterative improvement of a population of solutions (called individuals). As discussed later in the paper, several frameworks such as MOEA [20], MOMHLib++, Open BEAGLE [9], PISA [2], TEA [7] (to quote only them) already attempt to simplify and accelerate the development process of evolutionary MOO applications. We here propose a new library, called ParadisEO-MOEO (MOEO for short), that aims to produce efficient programs while having a minimal programming effort and a maximum code reuse. MOEO (Multi-Objective Evolving Objects) is an extension of the Evolving Objects framework [15]. It includes a broad range of reusable features and

techniques related to Pareto-based MOO such as performance metrics, elitism, fitness sharing and the most common Pareto-based fitness assignment schemes: MOGA, NSGA, NSGA-II, SPEA, SPEA2, IBEA, ... The fine-grained components of MOEO confer a high genericity, flexibility, adaptability and extensibility. Thus, a genuine conceptual effort has been done in order to allow the user to write only the minimum problem-specific code and to incrementally adapt an algorithm rather than entirely re-implementing it. Moreover, MOEO is itself extented to compose the full ParadisEO framework which is devoted to hybridization and parallel/distributed computing. Besides, MOEO has already been used to solve various academic problems likewise real-world applications.

The remainder of the paper is organized as follows. Section 2 gives the necessary background about MOO. Sections 3 and 4 describe the aims, the implementation and the provided features of the MOEO framework. In section 5, we present ParadisEO as well as the most common parallel/distributed models and hybridization mechanisms for multi-objective problems. In section 6, we survey some existing MOEO-designed applications and we introduce a Graphical User Interface for Multi-Objective Optimization (GUIMOO). Finally, the last section concludes the paper and highlights several perspectives about this work.

2 Multi-objective optimization

Widely investigated since the end of the 1980's, multi-objective optimization concerns many areas of the industry (telecommunication, transport, aeronautics, etc). In this section, we briefly present some required notions about Pareto-based multi-objective optimization such as the formulation of a multi-objective optimization problem (MOOP) and some concepts relating to Pareto optimality (the reader is referred to [4, 5] for more details).

Multi-objective optimization problem. A MOOP is defined by a decision space D, an objective space Z, and $n \geq 2$ objective functions $f_1, f_2, \ldots, f_n$. Each objective function can be either minimized or maximized. A solution $x = (x_1, x_2, \ldots, x_k)$ is represented by a vector of k decision variables. To each solution $x \in D$ is assigned exactly one objective vector $z \in Z$ on the basis of a vector function $F : X \rightarrow Z$ with $z = F(x) = (f_1(x), f_2(x), \ldots, f_n(x))$.

Pareto optimality. A multi-objective algorithm aims to approximate the set of Pareto optimal solutions according to F. A solution $x_a \in D$ is Pareto optimal if there exists no solution $x_b \in D$ that dominates x_a. For a minimization problem, the Pareto dominance relation is defined as follows:

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

The overall goal is then to find a well-converged and well-diversified set of Pareto optimal solutions.

These basic notions already emphasize the most important points to consider for the design of a library devoted to evolutionary multi-objective optimization.

3 ParadisEO-MOEO motivations

The 'EO' part of MOEO stands for *Evolving Objects*. EO is a C++ LGPL open-source object-oriented framework for evolutionary computation[1] that has been developed through an European joint work [15]. This library aims to provide a set of evolving objects dedicated to the flexible design of EAs. Furthermore, EO integrates many services including visualization facilities, on-line definition of parameters, application checkpointing, etc. MOEO is an extended version of the EO framework that includes some features related to Pareto-based multi-objective optimization. In this section, we present the goals of MOEO and we review some existing multi-objective optimization frameworks.

3.1 Goals

A framework is usually intended to be generic and could then be useful only if some important criteria are satisfied. Thence, the main goals of the MOEO framework are:

- *Services*. The framework must cover a wide range of features relating to Pareto-based multi-objective optimization.
- *Design and generic components*. MOEO must provide a whole architecture design of the solution method. This objective requires a clear and maximal conceptual distinction between the method and the problem representations. Therefore, the designer might only write the minimal problem-specific code, and the development process should be done in an incremental way.
- *Maximum code reuse*. The framework must allow the programmer to rewrite as little code as possible. Everthing that is already coded might be reusable. Then, it must be a commonplace to extend a problem from the mono-objective (and the EO framework) to the multi-objective case (and the MOEO framework), and from the classical to the parallel or the hybridizing case (and the whole version of the paradisEO framework, see section 5) without re-implementing the whole algorithm. For instance, it should not be necessary to re-code variation operators or solutions initialization.
- *Extensibility, flexibility and adaptability*. Some new features must easily be added or modified without implicating other components. Furthermore, existing components must be adaptable, as, in practice, existing problems evolve and new ones arise. Thence, MOEO must be a *white-box* framework (and not a black-box one); users must have access to source-code and must use inheritance or specialization to derive new components from base or abstract classes.

3.2 Existing multi-objective optimization frameworks

Many frameworks dedicated to combinatorial optimization have been proposed. However, very few reached the whole goals stated above. A non-exhaustive comparative study between some existing multi-objective optimization frameworks

[1] EO is available at http://eodev.sourceforge.net

is given in table 1. These frameworks are distinguished according to the following criteria: the available metaheuristic(s), the framework type (black-box or white-box), the licence (open-source or not), the available metrics, the availability of hybridization and parallel features and the programming language.

Table 1. Existing frameworks for multi-objective optimization (hybrid. stands for hybridization features, // for parallel features, lang. for programming language, ref. for reference, EA for Evolutionary Algorithm, LS for Local Search, SA for Simulated Annealing, TS for Tabu Search, ACO for Ant Colony Optimization and PSO for Particle Swarm Optimization).

name	available metaheuristic(s)	black box	open source	metrics	hybrid.	//	lang.	ref.
MOEA for Matlab	EA	X	X / -	-	-	X	Matlab	[20]
MOMHLib++	EA, LS, SA	-	X	R, coverage, O_w, O_s, O_c	X	-	C++	-
Open BEAGLE	EA	-	X	-	-	-	C++	[9]
PISA	EA	X	X	I_ϵ, $I_{\epsilon+}$, R_2, R_3, S	-	-	-	[2]
TEA	EA	-	X	-	-	X	C++	[7]
ParadisEO-MOEO	EA (+ LS, SA, TS)	-	X	$I_{\epsilon+}$, I_{HD}, entropy, contribution	X	X	C++	-

As we can see, the whole presented frameworks are open-source (only MOEA, although open-source, is based on Matlab which is not). Moreover, a large part of these frameworks are white-box frameworks, that is to say that the source-code can easily be extended or adapted in order to offer the most possible flexibility. Even so, only a thin part includes all the major metaheuristics and some metrics for performance evaluation or comparison. Also, parallel models and hybridization mechanisms are both provided at once only within ParadisEO-MOEO. Furthermore, ParadisEO is portable on distributed-memory machines and shared-memory multi-processors, it offers a high flexibility and, to our knowledge, is the only one that is portable on grid computing.

4 ParadisEO-MOEO implementation and deployment

Using EO and MOEO, it is possible to build a complete multi-objective evolutionary computation application. Two major contributions of the MOEO framework refer to i) archive-related features and ii) multi-objective fitness assignment techniques. On each level of its architecture, a set of classes, devoted to the first or the second point, is provided. First, the general implementation of a multi-objective EA is shown in order to see how simple is to code a whole algorithm

and to add or to change features. The implementation is conceptually divided into components so that different operators can be experimented without engendering significative modifications. A wide range of components are already provided, but new ones can easily be developed by the user with minimum code writing as MOEO is a white-box framework that tends to be flexible.

4.1 A general evolutionary algorithm implementation

Here is a general implementation of a multi-objective EA:

```
unsigned N;                          /* population size            */
eoPop<EOT> population;               /* population initialization  */
moeoeoArchive<EOT> archive;          /* archive declaration        */
eoEvalFunc<EOT> eval;                /* raw fitnesses evaluation   */
eoInit<EOT> init;                    /* solutions initialization   */
eoTransform<EOT> transform;          /* variation operators        */
eoContinue<EOT> stop;                /* stopping criteria          */
eoCheckPoint<EOT> checkpoint;        /* application checkpointing  */
eoPerf2Worth<EOT,double> p2w;        /* multi-objective ranking    */
eoSelectOne<EOT> selectOne;          /* selector (built using p2w) */
/* N-element selector */
eoSelect<EOT> select = eoSelectNumber<EOT>(selectOne, N);
eoReplacement<EOT> replace;          /* replacement                */
/* algorithm definition */
eoEasyEA<EOT> algo(stop, eval, select, transform, replace);
algo(population);                     /* run the algorithm         */
```

All evolution-related objects are templatized[2] regarding to the type of individuals (`EOT`). And, the `eoEasyEA` class is used to define the algorithm.

4.2 Archive-related features

An essential point of Pareto-based optimization is the concept of archive. An archive is a secondary population that stores non-dominated solutions. Its main goal is to prevent that these solutions are not lost during the (stochastic) optimization process. As a consequence, it must be updated at each generation with newly found non-dominated individuals:

```
moeoArchiveUpdater<EOT> updater (archive, pop);
checkpoint.add (updater);
```

Moreover, it is possible to save the fitnesses of the archive's members at each generation into a file `fileName` in order to study the evolution of the non-dominated set:

```
moeoArchiveFitnessSavingUpdater<EOT> fitness (archive, fileName);
checkpoint.add (fitness);
```

[2] A template is a generic description of a class or a function created as an instance of the template at compile time.

Performance metrics. Commonly, analyzing Pareto set approximations is done using performance metrics. The entropy [1] and the contribution [16] are both already provided within MOEO, but other ones can easily be implemented and a few will be soon. For instance, it is possible to save the progression of the entropy measured on the archive at every generation into a file `fileName`:

```
moeoEntropyMetric<EOT> entropy;
moeoBinaryMetricSavingUpdater<EOT>
                        metricUpdater (entropy, archive, fileName);
checkpoint.add (metricUpdater);
```

Elitist selection. Another major use of an archive is elitism [4, 5]. It consists in choosing individuals in the external population as well as in the current EA population during the selection phase of the algorithm, so that non-dominated solutions also contribute to the definition of variation operators.

```
eoSelectOne<EOT> popSelectOne;
eoSelectOne<EOT> archSelectOne;
selectOne = moeoSelectOneFromPopAndArch<EOT>
                 (popSelectOne, archSelectOne, archive, ratio);
```

At last, MOEO aims to constantly evolve and then, if need be, to provide further archive-related features in order to reflect the advances of the literature.

4.3 Implemented multi-objective fitness assignment strategies

In EO/MOEO, the fitness of a solution is represented by a vector of real numbers for which must be specified, for each criterion, if it is to be minimized or maximized. For multi-objective problems, fitness functions must convert raw fitnesses into fitness for selection. Various Pareto-based fitness assignment schemes are already implemented in EO and MOEO (see fig. 1), but this list is not exhaustive as the framework perpetually evolves and provides all that is necessary to easily implement new ones without a significant development effort.

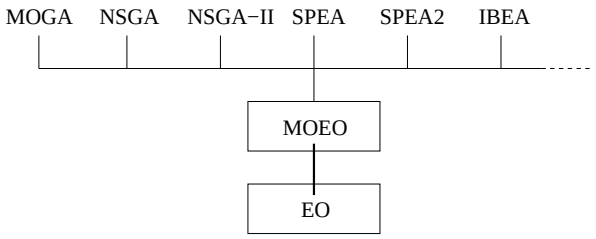

Fig. 1. Pareto-based multi-objective fitness assignment strategies proposed in EO and MOEO: MOGA, NSGA, NSGA-II, SPEA, SPEA2, IBEA, ...

Pareto-based fitness assignment was first proposed by Goldberg [10] to solve the problems of Schaffer's approach [17]. He suggested to use the Pareto dominance relation for ranking and selection. We here present all the fitness assignment strategies provided within MOEO as well as the code that is necessary to add or to modify in order to use them.

Pareto ranking (MOGA). In [8], Fonseca and Fleming proposed a variation of Goldberg's fitness assignment where a solution's rank corresponds to the number of solutions in the current population by which it is dominated (see fig. 2). Then, non-dominated individuals are all assigned the same rank, while dominated ones are penalized according to the population density in the corresponding region of the trade-off surface. The algorithm proceeds, first, by sorting the population according to the ranks previously determined. Then, fitness is assigned to solutions by interpolating from the best to the worst individuals in the population. Finally, fitnesses are averaged between solutions with the same rank.

```
eoDominanceMap<EOT> dominanceMap;
p2w = new eoParetoRanking<EOT> (dominanceMap);
```

Pareto sharing. As Goldberg and Deb noticed in [11], a fitness assignment like the previous one tends to produce premature convergence, what does not guarantee a uniformly sampled final Pareto approximation set. To avoid that, Fonseca and Fleming [8] modified the strategy above by implementing fitness sharing in the objective space to distribute the population over the Pareto-optimal region.

```
double nicheSize;
p2w = new moeoParetoSharing<EOT> (nicheSize);
```

NSGA. Srinivas and Deb [18] introduced another variation of Goldberg's fitness assigment in a similar way than [8], but based on Goldberg's version of Pareto-ranking. This algorithm, called *Non-dominated Sorting Genetic Alorithm*, classifies the solutions into several classes (or fronts). A solution that belongs to a class does not dominate another one from the same class. Then, individuals from the first front all belong to the best non-dominated set of the population; individuals from the second front all belong to the second best non-dominated set; and so on (see fig. 3). Logically, the best fitness value is assigned to solutions of the first class, because they are closest to the true Pareto-optimal front of the problem. This tends to search for solutions located in non-dominated regions. Additionally, a fitness sharing procedure helps to distribute the population over these regions.

```
double nicheSize;
p2w = new eoNDSorting_I<EOT> (nicheSize);
```

NSGA-II. In [6], Deb et al. introduced a modified version of NSGA. This new algorithm, called NSGA-II, is computationnaly more efficient, uses elitism, and keeps diversity without specifying any parameters by using a crowded tournament selection operator.

```
p2w = new eoNDSorting_II<EOT> ();
```

SPEA. Zitzler and Thiele [23] proposed an elitist algorithm called the *Strength Pareto Evolutionary Algorithm*. It maintains an external population (an archive) that stores a fixed number of non-dominated solutions found during the optimization process. For each member of the archive, a *strength value*, proportional to the number of solutions this member dominates, is computed. Then, the fitness of a solution is obtained according to the strength values of the archive's individuals that dominate it (see fig. 4). Moreover, a clustering method is used to keep diversity.

```
unsigned archiveSize;
select = moeoSPSelect_I<EOT> (N, archiveSize);
```

SPEA2. An improved version of SPEA, namely SPEA2, has been introduced by Zitzler et al. [22]. The three main differences of SPEA2 in comparison to its predecessor are that it incorporates: (i) a fine-grained fitness assignment strategy that takes into account the number of individuals that a solution dominates and is dominated by; (ii) a density estimation technique that leads the search process more precisely; (iii) an enhanced archive truncation method that ensures the preservation of boundary solutions.

```
unsigned archiveSize;
select = moeoSPSelect_II<EOT> (N, archiveSize);
```

IBEA. Introduced by Zitzler and Künzli [21], the *Indicator-Based Evolutionary Algorithm* (IBEA) has the characteristic to compute fitness values by comparing individuals on the basis of an arbitrary binary quality indicator I (also called binary performance metric). Thereby, no particular diversification mechanisms, such as fitness sharing, is necessary. The indicator, determined according to the decision maker preferences, denotes the overall goal of the optimization process. Thus, the fitness of a solution measures its usefulness according to the optimization goal. In MOEO, two binary quality indicators are proposed: the additive ϵ-indicator [24] and the I_{HD}-indicator [24] that is based on the hypervolume concept [23] (see fig. 5). However, everything is implemented to easily develop other indicators to be used with IBEA (see [24] for an overview about quality indicators).

```
moeoSolutionVsSolutionBM<EOT> I;
double kappa;       /* scaling factor */
p2w = new moeoIBSorting<EOT> (I, kappa);
```

New fitness assignment strategies. MOEO aims to be extensible, flexible and easily adaptable. All its components are generic in order to provide a modular architecture design that allows the user to quickly and conveniently develop a new fitness assignment scheme with a minimum code writing. The aim is to follow the new strategies coming from the litterature and, if need be, to provide any additional components required for their implementation.

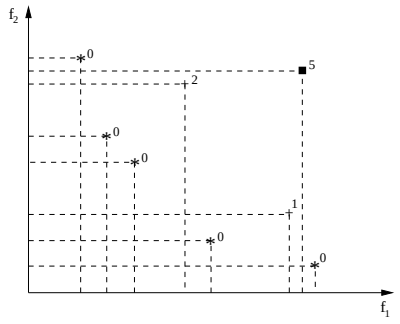

Fig. 2. The Pareto ranking fitness assignment scheme on a two-objective minimization problem.

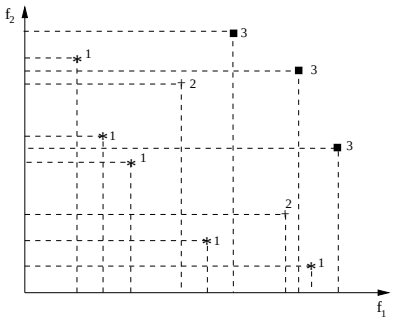

Fig. 3. The NSGA fitness assignment scheme on a two-objective minimization problem.

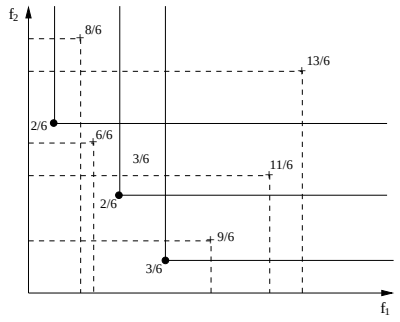

Fig. 4. The SPEA fitness assignment scheme on a two-objective minimization problem (the external population members are shown by circles and the EA population members are shown by crosses).

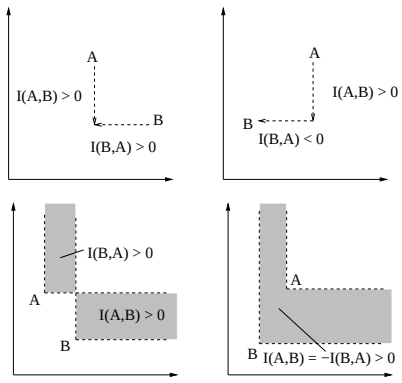

Fig. 5. Illustration of binary quality indicators for a two-objective minimization problem (top: $I_{\epsilon+}$-indicator, bottom: I_{HD}-indicator).

5 Parallelism and hybridization design for multi-objective problems using the ParadisEO framework

In practice, multi-objective optimization problems are varied, they perpetually evolve (with regards to the needs, the constraints, the objectives, etc), they handle a high number of decision variables and they have to deal with instances of increasing size. Despite that, the overall goal is still to find near Pareto-optimal solutions in a tractable time. Then, classical approaches are not sufficient, and hybridization features as well as large scale parallelism must be considered to tackle this kind of problem. As shown in figure 6, in addition to parallel and distributed environments, the ParadisEO framework embeds a broad range of features, including evolutionary algorithms (as it is based on the EO framework), various local searches (as it is based on the MO framework) and multi-objective mechanisms (as it is based on the MOEO framework). Furthermore, its generic aspect allows to easily add some of its features to a MOEO-designed problem.

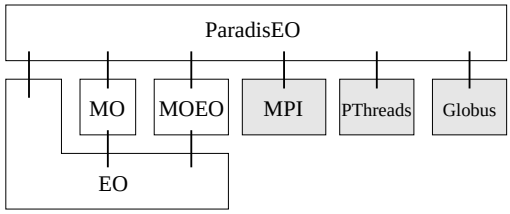

Fig. 6. ParadisEO architecture: Evolving Objects (EO) for the design of population-based metaheuristics, Moving Objects (MO) for the design of solution-based metaheuristics and their hybridization with EAs, Multi-Objective EO (MOEO) for multi-objective optimization, and ParadisEO for parallel and distributed models. These models are portable on many execution platforms thanks to the MPI, PThreads and Globus standard libraries.

5.1 Parallel distributed evolutionary algorithms

Basically, three major parallel models can be distinguished [3]: the cooperative island model, the parallel population evaluation model, and the distributed single-solution evaluation model. These models are all illustrated in figure 7.

The cooperative island model. A number of EAs are simultaneously deployed to cooperate with the aim of improving the solutions's robustness. Each of them performs a search on a sub-population. Then, exchanges of genetic materials are performed in an asynchronous way to diversify the search into the target sub-populations. This allows to delay the global convergence, especially when the EAs are heterogeneous with respect to the variation operators. Individuals migrations are conducted by various parameters and are performed in a regular or irregular way.

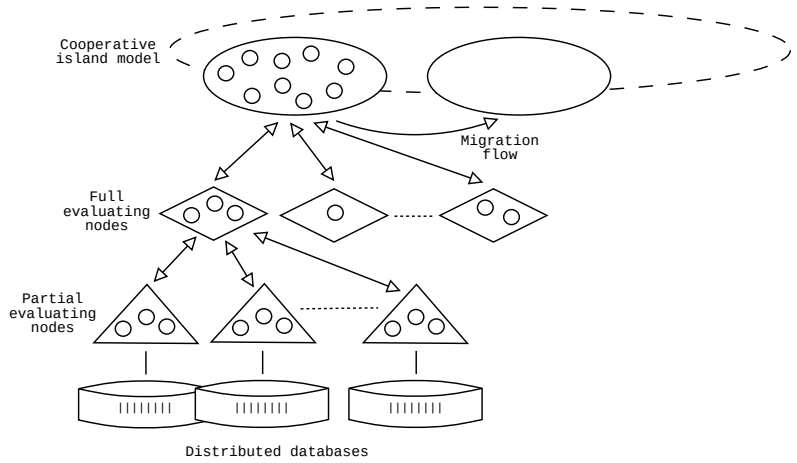

Fig. 7. Three major parallel models for multi-objective EAs.

The parallel population evaluation model. The evaluation step of an EA is generally the most time-consuming. Therefore, in order to speed up the search, this centralized parallel model distributes the evaluation of the evolving population. As they require a global management of the population, the selection, the transformation and the replacement operations are applied by a master process. At each generation, it distributes the set of newly generated solutions between different workers that evaluate and return back these solutions as well as their fitness values. A particularly efficient execution is often obtained when the ratio between communication and computation is high.

The distributed single-solution evaluation model. The fitness of each solution is evaluated in a parallel centralized way. Such a model is especially interesting when the evaluation of a solution can be itself parallelized as it requires, for instance, an access to large databases distributed among various processing nodes.

Those three parallel and distributed models are all provided within ParadisEO [3]. They are implemented using MPI, PThreads and Globus standard libraries, and are thus portable on different execution platforms such as parallel computing, cluster computing, internet computing and grid computing. Moreover, their deployement is transparent as the user does not need to manage the communications and threads-based concurrency.

5.2 Hybridization

Hybridization have acquired a considerable interest in the field of optimization these last years [19]. A wide variety of hybrid approaches exists in the literature. And, for many academic and real-world applications, best found solutions are obtained by hybrid algorithms. In the multi-objective context, EAs are generally

hybridized with local search methods in order to apply the local search algorithm on a selected individual, or to find non-dominated solutions in the neighborhood of an interesting region of the objective space. In [19], two levels (low and high) and two modes (relay and cooperative) of hybridization are distinguished.

The low-level hybrid algorithms address the functional composition of a single optimization method. A given function of a metaheuristic is replaced by another metaheuristic. For high-level hybridization, the different metaheuristics are self-contained. There is no direct relationship between the internal workings of a metaheuristic. Besides, for relay hybridization, a set of metaheuristics is applied the one after another in a pipeline way, each one using the output of the previous one as its input. Contrarily, cooperative hybridization represents a teamwork optimization model in which parallel cooperating agents perform a search in a solution space and exchange solutions with the others.

The ParadisEO framework provides all these most common hybridization mechanisms [3] that can thus directly be applied to a MOEO-designed application in a fast and simple way. They can naturally be exploited to make cooperating metaheuristics belonging either to the same or to different families.

6 Applications

ParadisEO and MOEO have been applied to many areas where multi-objective optimization is required. Before presenting three examples of applications drawn from varied fields that have been implemented within MOEO, we will introduce some prerequisites concerning performance evaluation and results analysis.

6.1 Preliminaries: GUIMOO

In multi-objective optimization, a fundamental part is the performance comparison of various algorithms. Therefore, the question arises on the way of evaluating the quality of Pareto set approximations. To achieve that, let us introduce GUIMOO[3] (a Graphical User Interface for Multi-Objective Optimization). This platform-independant free software is dedicated to the analysis of results for multi-objective optimization and is able to handle different input and output formats. Its main features are:

- The on-line and off-line visualization (in 2 or 3 dimensions) of Pareto set approximations. A Pareto set approximation can be characterized by its (dis)continuity, (dis)convexity, multi-modality, ... Such an information can be useful to help an expert to build more efficient metaheuristics.
- A number of metrics for quantitative and qualitative performance evaluation or comparison [24] (contribution, entropy, generational distance, spacing, coverage of two sets, coverage difference, S-metric, D-metric and R-metrics).

Furthermore, GUIMOO aims to be generic. Its architecture allows to easily customize it in order to provide more functionalities to tackle specific applications (telecom, genomics, engineering design, etc).

[3] GUIMOO is available at http://guimoo.gforge.inria.fr

6.2 Examples

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

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

Electromagnetic properties of conducting polymer composites in the microwave band. Due to the proliferation of electromagnetic interferences, designing protecting material for high frequencies equipments has become an important problem. In [14], a new multi-objective model is proposed to design the different layers of a conducting polymer. To solve this model, a multi-objective continuous genetic algorithm is used. This algorithm offers several solutions with different physical properties and different costs.

Knowledge discovery in biological data from microarray experiments. The problem of analyzing microarray data is actually a major issue in genomics. Often used techniques are clustering and classification. In [13], the authors propose to analyze those data through association rules. The problem is modeled as a multi-objective rule mining problem and a genetic algorithm is used to explore the large search space associated. Thence, MOGA permitted to present previously undiscovered knowledge.

7 Conclusion and perspectives

In this paper, we introduced ParadisEO-MOEO, a framework dedicated to the reusable design of evolutionary multi-objective optimization applications[4]. It provides the most common Pareto-based multi-objective fitness assignment shemes (MOGA, NSGA, NSGA-II, SPEA, SPEA2, IBEA, ...) as well as fitness sharing and a wide range of archive-related features such as non-dominated solutions storage, elitism and performance metrics computation. Moreover, the whole version of ParadisEO, a complete framework for the design of parallel/distributed and hybrid metaheuristics, and GUIMOO, a software for the analysis and the comparison of Pareto set approximations, have both been presented. These frameworks have all been applied to many type of applications, from academic to real-world problems.

ParadisEO-MOEO is an open-source white-box object-oriented framework that aims to simplify and speed up the incremental implementation of a whole

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

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

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

References

1. Basseur M., Seynhaeve F., Talbi E.-G.: Design of multi-objective evolutionary algorithms: Application to the flow-shop scheduling problem. In Congress on Evolutionary Computation (CEC'02), Honolulu, Hawaii, USA (2002) 1151–1156
2. Bleuler S., Laumanns M., Thiele L., Zitzler E.: PISA — A Platform and Programming Language Independent Interface for Search Algorithms. In Fonseca C. M. et al. (eds.): Evolutionary Multi-Criterion Optimization, Second International Conference (EMO 2003). Lecture Notes in Computer Science **2632**, Springer, Faro, Portugal (2003) 494–508
3. Cahon S., Melab N., Talbi E.-G.: ParadisEO: A Framework for the Reusable Design of Parallel and Distributed Metaheuristics. Journal of Heuristics **10(3)** (2004) 357–380
4. Coello Coello C. A., Van Veldhuizen D. A., Lamont G. B.: Evolutionary Algorithms for Solving Multi-Objective Optimization Problems. Kluwer Academic Publishers, New York, USA (2002)
5. Deb K.: Multi-Objective Optimization using Evolutionary Algorithms. Wiley, Chichester, UK (2001)
6. Deb K., Agrawal S., Pratap A., Meyarivan T. A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. In Proc. of the 6th International Conference on Parallel Problem Solving from Nature (PPSN VI), Paris, France (2000) 849–858
7. Emmerich M., Hosenberg R.: TEA - A Toolbox for the Design of Parallel Evolutionary Algorithms in C++. Technical report CI-106/01, SFB 531, University of Dortmund, Germany (2001)

8. Fonseca C. M., Fleming P. J.: Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization. In Forrest S. (ed.): Proc. of the 5th International Conference on Genetic Algorithms, Morgan Kaufmann publishers, San Mateo, California (1993) 416–423
9. Gagné C., Parizeau M.: Genericity in Evolutionary Computation Software Tools: Principles and Case Study. International Journal on Artificial Intelligence Tools **15(2)** (2006) 173–194
10. Goldberg D. E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley, Boston, MA, USA (1989)
11. Goldberg D. E., Deb K.: A Comparative Analysis of Selection Schemes Used in Genetic Algorithms. In Rawlins G. (ed.): Foundations of Genetic Algorithms, Morgan Kaufmann, San Mateo, CA, USA (1991) 69–93
12. Holland J. H.: Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Abor, MI, USA (1975)
13. Jourdan L., Khabzaoui M., Dhaenens C., Talbi E.-G.: A Hybrid Evolutionary Algorithm for Knowledge Discovery in Microarray Experiments. In Olariu S. et al. (eds.): Handbook of Bioinspired Algorithms and Applications, CRC Press, USA, chapter 28 (2005) 489–505
14. Jourdan L., Legrand T., Talbi E.-G., Wojkiewicz, J.-L.: Mono and Multi-objective continuous optimization for conducting polymer composites. ECCO/CO 2006, Porto, Portugal (2006)
15. Keijzer M., Merelo J.-J., Romero G., Schoenauer M.: Evolving Objects: a general purpose evolutionary computation library. In Proc. of the 5th International Conference on Artificial Evolution (EA'01), Le Creusot, France (2001) 231–244
16. Meunier H., Talbi E.-G., Reininger P.: A multiobjective genetic algorithm for radio network optimization. In Proc. of the 2000 Congress on Evolutionary Computation (CEC'00). IEEE Press (2000) 317-324
17. Schaffer J. D.: Multiple Objective Optimization with Vector Evaluated Genetic Algorithms. In Grefensette J. J. (eds.): Proc. of the 1st International Conference on Genetic Algorithms, Pittsburgh, PA, USA (1985) 93–100
18. Srinivas N., Deb K.: Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. Evolutionary Computation **2(3)** (1994) 221–248
19. Talbi E.-G.: A Taxonomy of Hybrid Metaheuristics. Journal of Heuristics **8(5)** (2002) 541–564
20. Tan K. C., Lee T. H., Khoo D., Khor E. F., Kannan R. S.: MOEA Toolbox for Computer Aided Multi-Objective Optimization. In Proc. of the 2000 Congress on Evolutionary Computation (CEC'00). IEEE Press (2000) 38–45
21. Zitzler E., Künzli S.: Indicator-Based Selection in Multiobjective Search. In Parallel Problem Solving from Nature (PPSN VIII). Lecture Notes in Computer Science **3242**, Springer, Birmingham, UK (2004) 832–842
22. Zitzler E., Laumanns M., Thiele L.: SPEA2: Improving the Strength Pareto Evolutionary Algorithm. TIK Report Nr. 103, Computer Engineering and Networks Lab (TIK), Swiss Federal Institute of Technology (ETH) Zurich, (2001)
23. Zitzler E., Thiele L.: Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. IEEE Transactions on Evolutionary Computation **3(4)** (1999) 257–271
24. Zitzler E., Thiele L., Laumanns M., Fonseca C. M., Grunert da Fonseca V.: Performance Assessment of Multiobjective Optimizers: An Analysis and Review. IEEE Transactions on Evolutionary Computation **7(2)** (2003) 117–132