# Evolutionary Multi-Criterion Optimisation

Carlos M. Fonseca
CSI – Centro de Sistemas Inteligentes
Faculdade de Ciências e Tecnologia, Universidade do Algarve
Campus de Gambelas, 8005-139 FARO, Portugal
e-mail: cmfonsec@ualg.pt

**Abstract**

Evolutionary algorithms, a broad class of optimisation algorithms inspired in the process of natural evolution, are introduced, and an artificial model of evolution is given which encompasses most established evolutionary algorithm variants. This model is then reinterpreted in the light of multiobjective optimisation, and a link to decision analysis is established.

## 1 Introduction

Nature has been a major source of inspiration and metaphors for scientific and technical development. It is not difficult to identify the links between the ear and the microphone, the eye and the camera, the bat and the sonar system, the brain and artificial neural networks, and so forth. Similarly, the process of natural evolution has inspired a growing amount of research in artificial systems, only made possible by the increasing availability of modern computing power.

Evolutionary optimisation is a term used to describe a broad class of optimisation algorithms inspired in the process of natural evolution. Such evolutionary algorithms (EAs) have been applied with various degrees of success to many difficult optimisation problems in engineering and operations research, among several other areas.

Many practical optimisation problems involve, not a single objective, but a number of possibly competing objectives, or criteria. Although different objectives may usually be combined by means of aggregating functions to produce a single cost measure, it is not always easy, or even appropriate, to define such a function. In the absence of information about the relative importance of individual objectives, the optimisation problem may still be approached, but will generally admit no unique solution. Rather, a number of optimal solutions may exist, in the sense that each such solution may be improved with respect to some criteria only at the expense of degradation in other criteria.

In this paper, evolutionary optimisation is introduced in the more general context of evolutionary processes, after reviewing some relevant problem solving concepts. Then, an artificial model of evolution is given which encompasses most established EA variants. Finally, that model is reinterpreted so as to accommodate multiple criteria, and a link to decision analysis is established.

## 2 Problem solving

Evolutionary algorithms may be defined as a broad class of computational methods inspired in the process of natural evolution, which are aimed at solving difficult problems. Before considering EAs in more detail, it is worth reviewing some concepts related to problem solving.

## 2.1 What is a problem?

**Definition 1 (Abstract problem)** *An* abstract problem *$Q$ is a binary relation on a set $I$ of problem* instances *and a set $S$ of problem* solutions *[1].*

Considering the Travelling Salesman Problem (TSP) as an example, an *instance* consists of a set of cities and of the distances between them. A *solution*, on the other hand, consists of a sequence of cities, which describes the order in which they should be visited. Note that this view of a problem is very general, and that one is often interested in more restricted classes of problems.

**Definition 2 (Decision problem)** *An abstract problem is called a* decision problem *if it has a* yes/no *solution, i.e. $S = \{0,1\}$ [1].*

**Definition 3 (Optimisation problem)** *An abstract problem is called an* optimisation problem *if it consists of finding* minimal *or* maximal *elements [2] of a set $S$ under a given* preorder $\preceq$.

Returning to the previous example, the TSP is an optimisation problem, as it consists of finding a tour of minimum length. Here, the preorder $\preceq$ on $S$, the set of all possible tours, may be defined by referring to the tour length as a cost function $f$ of each city sequence $x \in S$:

$$\forall x_1, x_2 \in S,\ x_1 \preceq x_2 \Leftrightarrow f(x_1) \leq f(x_2)$$

Note also that, given a TSP, the problem of whether or not a tour exists which is shorter than a given length (or than a given tour) is a decision problem. One may attempt to solve such a decision problem by considering a candidate solution to the original optimisation problem, $x \in S$, and *verifying* whether its cost $f(x)$ is indeed less than or equal to the given bound. If so, the decision problem is know to have an affirmative solution. Otherwise, it remains unsolved.

In general, optimisation problems can be recast as decision problems in the way just described.

## 2.2 Encodings

The difficulty of a problem is inherently related to the time needed to solve it, regardless of its type. One important issue in discussing problem difficulty is that, to be solved on a computer, an abstract problem instance must be represented in some way.

**Definition 4 (Encoding)** *An* encoding *of a set $S$ of abstract objects is a mapping $e$ from $S$ to the set of binary strings [1].*

**Definition 5 (Concrete problem)** *A problem is called a* concrete problem *if its instance set $I$ is the set of binary strings [1].*

It is important to note that the time taken by a computer to solve a concrete problem my depend heavily on the underlying encoding. Thus, complexity theory restricts itself to concrete problems, and concrete decision problems in particular. Although complexity theory will not be discussed further here, one should realise that if an optimisation problem can be solved quickly, then so can any associated decision problem. Equivalently, if a decision problem is hard to solve, the related optimisation problem will also be hard.

As it will be discussed later, encodings may play a very important role in evolutionary optimisation.

### 2.3 Evolutionary algorithms as approximation algorithms

When a given optimisation problem cannot be solved exactly in an acceptable amount of time, it may still be possible to find approximate solutions, e.g. by verifying given candidate solutions against the best solution known to date. Indeed, EAs act as optimisers by

1. generating candidate solutions
2. evaluating them
3. using the information thus gained to generate new, possibly better, candidate solutions

Together with methods such as *Tabu Search*, *Simulated Annealing*, *Stochastic Local Search*, and *Ant Colony Optimisation*, among others, EAs integrate a class of approximation techniques which has become known as *Metaheuristics*.

## 3 Evolutionary processes

The process of natural evolution has inspired a growing amount of research in artificial systems. The resulting class of computational methods which simulate various aspects of natural evolution became known as Evolutionary Algorithms, having attracted interest from biology, chemistry, economics, engineering and mathematics. The area emerged in the late 1960s in Europe [3, 4] and the US [5, 6], motivated by a desire to advance the state of the art in optimisation, adaptation, and machine learning. It became popular in the 1990s due, to a great extent, to the publication of Goldberg's book [7], and has continued to grow since then.

As an optimisation process, evolution has many interesting features. In particular, individuals are selected based on how well they function, and not based on the mechanisms which account for their functionality. Thus, they tolerate a limited understanding of how existing solutions may be improved, allowing problems previously considered intractable to be approached.

*Neo-Darwinism* is currently the most widely accepted paradigm of natural evolutionary [8]. It is based on the four essential processes of reproduction, mutation, competition and selection. Reproduction consists of individuals being capable of generating offspring similar to themselves (either sexually or asexually), whereas mutation accounts for replication errors during reproduction. Competition arises as the number of individuals in a population grows in a resource-limited environment, and selection consists of only certain individuals, through competition, actually being able to reproduce. Since offspring tend to be similar to their parents, selection effectively shapes populations as they evolve.

### 3.1 Population

In the neo-Darwinian paradigm, individuals can be understood as a duality. The genetic programme, or *genotype*, consists of an *encoded* representation of the individual at the chromosome level. Individual traits, on the other hand, are expressed by executing, or decoding, the genotype. Expressed traits are also called the *phenotype*, and usually vary as a complex non-linear function of the genotype, and of its interaction with the environment. In particular, there are usually no one-gene one-trait relationships. A single gene may simultaneously affect many phenotypic traits (*pleiotropy*) and a single phenotypic trait may be affected by the interaction of several *genes* (*polygeny*).

In artificial evolutionary systems, however, the encodings used are usually simple and concise, and seldom implement pleiotropy. Polygeny usually occurs, though, especially due to the lower cardinality of the alphabets tend to be used to encode the genotype.

## 3.2 Selection

Individuals are selected based on their expressed traits. Selection determines the survival of the best individuals and, consequently, their opportunity to generate offspring. Individuals which produce more offspring are considered *fitter* than others. Indeed, in natural systems, fitness is *expressed*: individuals are fit *because* they generate offspring. On the contrary, in artificial evolutionary systems, fitness is usually *assigned* to individuals based on some criterion, e.g., the cost function which defines an optimisation problem. This is perhaps one of the fundamental differences between natural evolution and evolutionary optimisation. The reproductive advantage of the best individual in a population with respect to the population's average is known as *selective pressure*.

Selection may be implemented in several ways. In *generational* selection, individuals reproduce all at the same time, and offspring replace the whole of the parent population, never competing with it. This is akin to the reproductive cycle of some insects, for example, where parents die before offspring are born. In an alternative model of selection, parents may be selected to reproduce at any time, and the offspring they generate are inserted in the parent population and forced to compete with it (*incremental* selection).

Another aspect of selection is whether it is *stochastic*, as it is common in natural systems, or *deterministic*, as in animal breeding, for example. One interesting property of (stochastic) selection, which can be observed in Nature as well as on the computer, is known as *genetic drift*, and consists of the tendency finite populations exhibit to evolve towards a single type of solution even when equivalent alternatives exist [9]. On the computer, genetic drift may be controlled in some circumstances by implementing *niche induction* mechanisms [10] such as *fitness sharing* and *crowding*.

## 3.3 Heredity

Evolution does not arise out of selection alone. One fundamental aspect of evolution is that individual replication is not perfect, i.e., individuals are not exactly like their parents but differ from them to a certain extent. As reproduction occurs at the genotypic level, the basic assumptions are that:

1. Offspring are similar to their parents at the genotypic level (heredity).

2. Good individuals have similar genotypes.

On the computer, as in Nature, heredity may assume several forms:

**Mutation** Random alteration of only small parts of individual genotypes

**Recombination** Production of offspring from the genotypic material of two (or possibly more) parents

**Learning** Incorporation of knowledge acquired at higher levels into the an individual's representation. This may be seen as *Lamarckian* evolution, *memetic* evolution, or even as "genetic engineering".

Despite the fact that there can be no evolution without variation, variation must be controlled. The basic idea is that selection must be able to recover from any deleterious variation. The amount of variation beyond which evolution no longer occurs is known as the error threshold [11].

```
Evolutionary Algorithm:
  1. Generate initial population (genotypes)
  2. While not happy do
       (a) decode individuals (phenotypes)
       (b) evaluate individuals (cost)
       (c) assign reproductive ability (fitness)
       (d) select individuals based on fitness
       (e) produce offspring (genotypes)
       (f) insert offspring into the population
```

Figure 1: An artificial model of evolution.

### 3.4 Viability

Finally, it is not sufficient either for good individuals to produce many offspring, as their offspring must be fit as well for evolution to successfully occur. Whether or not this is the case depends on the optimisation problem itself, on the genotypic encoding and on the variation mechanisms which manipulate it. Electing a good combination of encoding and variation operators for a given problem continues to be perhaps the greatest challenge in evolutionary optimiser design.

## 4 An artificial model of evolution

The various concepts introduced above can now be combined to produce a general artificial model of evolution, as depicted in Figure 1.

In an evolutionary optimisation setting, the initial population is typically drawn at random from a suitable encoding of the solution set $S$. Encodings commonly found in the literature include binary strings, $n$-ary strings, permutations, graphs (and especially trees), and combinations of these, depending on the problem considered.

Individual genotypes are then decoded to yield candidate solutions in (a subset of) the solution space $S$. Encodings are usually such that each genotype typically decodes into a unique phenotype, although it is possible to consider encodings where this is not always the case [12].

### 4.1 Evaluation and fitness assignment

Once candidate solutions in $S$ have been obtained, individuals are evaluated based on the preorder which defines the optimisation problem or on a cost function, if one is given. Provided that the preorder $\preceq$ is such that all individuals are comparable (i.e., $\forall a,b \in S$, $a \preceq b \vee b \preceq a$), evaluation may consist of no more than sorting the population and noting individual ranks.[13]). Alternatively, the cost function may be evaluated at each individual. This is the usual single-objective optimisation case.

Evaluated solutions are then assigned a fitness value. Fitness may be assigned based on rank (*ranking* [13]) or as a function of cost function values (*scaling*). Scaling is the more traditional approach. Raw fitness is calculated as a monotonic function of the cost, offset by a certain amount, and then linearly scaled. The first difficulty arises at this stage: whilst scaling aims to preserve the relative performance between different individuals, both the initial transformation and the subsequent offsetting can significantly affect the fitness ultimately assigned to each individual.

With scaling, an individual much stronger than all the others may be assigned a very large fitness and, through selection, rapidly dominate the population. Conversely, the advantage of the best individual over the rest of the population will be minimal if most individuals perform more or less equally well, and the search will degenerate into an aimless walk.

Ranking addresses these difficulties by eliminating any sensitivity to the scale in which the problem is formulated. Since the best individual in the population is always assigned the same fitness, would-be "super" individuals can never reproduce excessively. Similarly, when all individuals perform almost equally well, the best individual is still unequivocally preferred to the rest (but this may be inappropriate if the objective function is contaminated with noise).

Rank-based fitness assignment is characterised by the choice of rank-to-fitness mapping, which is usually chosen to be linear or exponential. For a population of size $N$, ranking the best individual zero and the worst $N-1$, and representing rank by $r$ and fitness by $\phi(r)$, these mappings can be written as follows:

**Linear**

$$\phi(r) = s - (s-1) \cdot \frac{2r}{N-1},$$

where $s$, $1 < s \leq 2$, is the fitness desired for the best individual. The upper bound on $s$ arises because fitness must be non-negative for all individuals, while maintaining $\sum_{i=0}^{N-1} \phi(i) = N$.

**Exponential**

$$\phi(r) = \rho^r \cdot s,$$

where $s > 1$ is the fitness desired for the best individual, and $\rho$ is such that $\sum_{i=0}^{N-1} \rho^i = N/s$. Since there is no upper-bound on $s$, the exponential mapping is somewhat more flexible than the linear.

For $1 < s \leq 2$, the main difference between linear and exponential rank-based fitness assignment is that the exponential mapping does not penalise the worst individuals as much, at the expense of assigning middle individuals fitness slightly less than average. As a consequence, exponential assignment generally contributes to a more diverse search.

## 4.2 Selection

A number of parents are selected from the population through a sampling mechanism, which may be deterministic or stochastic. A well-established sampling procedure is known as Stochastic Universal Sampling [14], and may be visualised as the result of spinning a roulette wheel with slots proportional in width to the fitness of the individuals in the population, and with multiple, equally spaced pointers (Figure 2). Once it stops, the number of pointers over each sector must be an integer, either immediately above or immediately below the corresponding desired number of offspring, guaranteeing minimum deviations from the desired fitness value. The replicates obtained in this way should be shuffled before the algorithm proceeds with recombination.

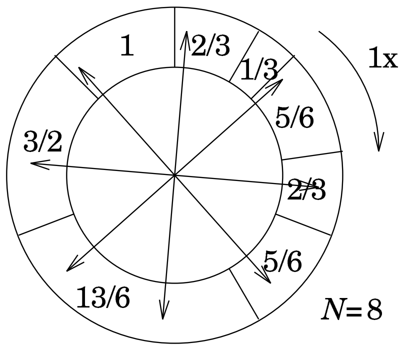

Figure 2: Stochastic Universal Sampling

### 4.3 Recombination and mutation

Offspring are produced from the parents selected, by manipulating them at the genotypic level. Parents may be recombined and/or mutated to generate offspring. A typical recombination operator for binary and other string chromosomes is single-point crossover, whereby two individuals exchange a portion (right or left) of their chromosomes to produce offspring, as illustrated in Figure 3. The crossover point is selected at random. Other recombination operators commonly used with binary strings are:

**Double-point crossover** Two crossover points are selected instead of one [15].

**Uniform crossover** Each bit is exchanged independently, with a given probability [16].

**Shuffle crossover** The chromosomes are shuffled before single-point crossover is applied, and consequently deshuffled [17].

**Reduced-surrogate crossover** The non-identical bits in the chromosomes are first identified, and one of the above crossover types applied to the smaller string thus defined [15]. This has the effect of guaranteeing the production of offspring different from their parents.

As for bit mutation (see Figure 4), it is most commonly implemented by independently flipping each bit in the chromosome with a given probability.

### 4.4 Reinsertion

Finally, the offspring produced are inserted in the population, replacing:

- random members of the parental population,
- the oldest members of the parental population,
- their own parents, or
- the least fit members of the parental population.

Actual reinsertion may occur

- unconditionally,

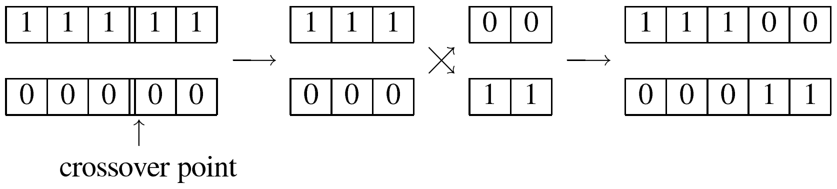

Figure 3: Single point crossover

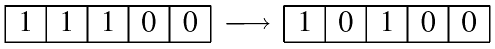

Figure 4: Bit mutation

- only if the offspring are fitter than the individuals they are to replace, or
- probabilistically, depending on whether or not the offspring are stronger than the individuals they are to replace.

Note that, by denying some individuals the possibility of reproducing further, reinsertion has ultimately the same effect as selection. The overall *selective pressure* imposed on the population is not only determined by the fitness assignment strategy, but is also affected by when and how reinsertion is performed. In particular, always replacing the least fit individuals in the population strongly increases the *effective*, as opposed to *assigned*, fitness differential between stronger and weaker individuals in the population. This is because, in addition to being less likely to be selected, weaker individuals tend to die earlier, thus participating in less selection trials than stronger ones. Reinsertion strategies which guarantee the preservation of the best individual are known as *elitist*.

## 5 Multiobjective optimisation

Practical problems are often characterised by several non-commensurable and often competing measures of performance, or objectives. The multiobjective optimisation problem may be stated as the problem of simultaneously minimising the $n$ components $f_i$, $i = 1,\dots,n$, of a vector function $\mathbf{f}(x)$, with $x \in S$, where

$$\mathbf{f}(x) = (f_1(x),\dots,f_n(x)).$$

The problem usually has no unique, perfect (or Utopian) solution, but may admit a set of non-dominated, alternative solutions, known as the Pareto-optimal set [18]. Assuming a minimisation problem, dominance is defined as follows:

**Definition 6 (Pareto dominance)** *A real vector* $\mathbf{u} = (u_1,\dots,u_n)$ *is said to dominate* $\mathbf{v} = (v_1,\dots,v_n)$ *if and only if* $\mathbf{u}$ *is partially less than* $\mathbf{v}$ *(*$\mathbf{u}\ {}_{\mathrm{p}}\!< \mathbf{v}$*), i.e.,*

$$\forall i \in \{1,\dots,n\}\ ,\ u_i \le v_i \quad \wedge \quad \exists i \in \{1,\dots,n\} :\ u_i < v_i\,.$$

**Definition 7 (Pareto optimality)** *A solution* $x_{\mathrm{u}} \in S$ *is said to be Pareto-optimal if and only if there is no* $x_{\mathrm{v}} \in S$ *for which* $\mathbf{v} = \mathbf{f}(x_{\mathrm{v}}) = (v_1,\dots,v_n)$ dominates $\mathbf{u} = \mathbf{f}(x_{\mathrm{u}}) = (u_1,\dots,u_n)$.

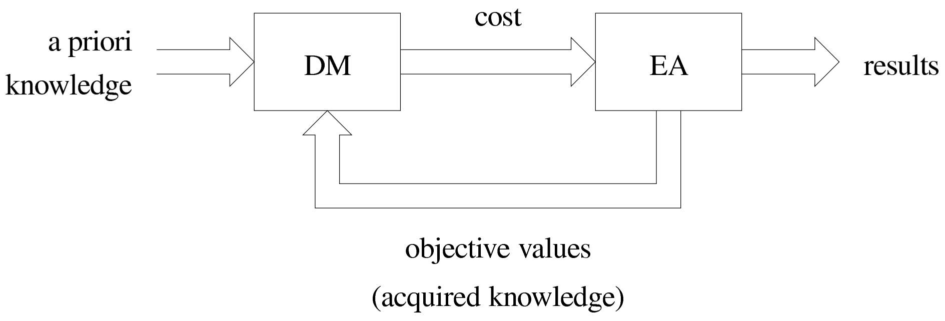

Figure 5: A general multiobjective evolutionary optimiser.

Pareto-optimal solutions are also called efficient, non-dominated, and non-inferior solutions. The corresponding objective vectors are simply called non-dominated. The set of all non-dominated vectors is known as the non-dominated set, or the trade-off surface, of the problem.

Alternatively, the multiobjective optimisation problem may be defined by specifying the preorder in Definition 3 as:

$$\forall x_1, x_2 \in S, \quad x_1 \preceq x_2 \Leftrightarrow f_i(x_1) \leq f_i(x_2)\ \forall i \in \{1, \ldots, n\}.$$

This greatly simplifies establishing a formulation of multiobjective evolutionary algorithms.

## 5.1 Multiobjective evolutionary algorithms

Definition 3 is general enough that it accommodates both single and multiobjective optimisation problems. On the other hand, the artificial model of evolution presented in Figure 1 was based simply on this definition. It is therefore clear that the main difference between a single-objective and a multiobjective evolutionary algorithm must lie in the individual evaluation step. In particular, some elements of $S$ may now be incomparable and assigning a cost value to each candidate solution becomes a *decision analysis* problem.

A general multiobjective evolutionary optimiser may also be seen as the result of the interaction between between a Decision Maker (DM) and an Evolutionary Algorithm (see Figure 5). The EA generates a new set of candidate solutions according to the cost assigned to the current set of candidates by the DM. New candidate solutions, as they are evaluated provide new trade-off information which the DM can use to refine the current preferences. The EA sees the effect of any changes in the decision process, which may or may not result from taking recently acquired information into account, as an environmental change. The DM block represents any cost assignment strategy, which may range from that of an intelligent Decision Maker to a simple aggregating function approach.

Aggregating function approaches to multiobjective evolutionary optimisation, although useful and very common in the literature, do convert multiobjective optimisation problems into a single-objective problems, raising no particular issues as far as the EA formulation is concerned. A number of alternative approaches, known as population-based approaches [19], typically assign different objectives to different subsets of the population, so as to promote the emergence of good compromise solutions. Schaffer's pioneering work on *Vector Evaluated Genetic Algorithms* [20] falls in this category. A third class of approaches is based directly on the definition of Pareto-dominance, and includes most modern evolutionary multiobjective optimisers.

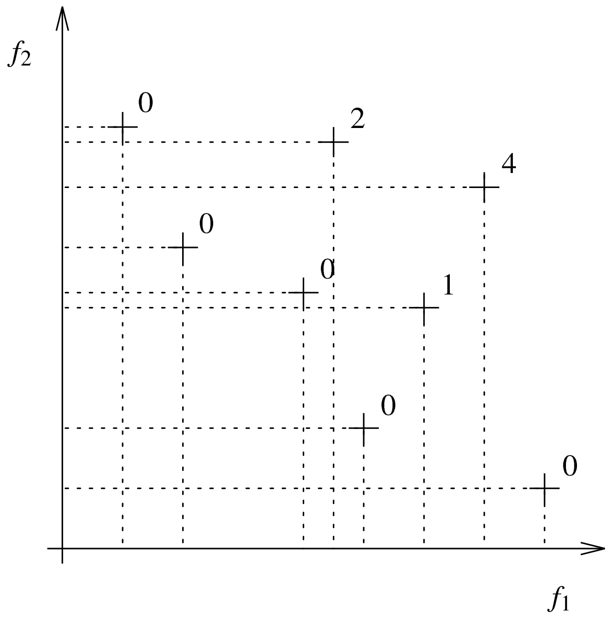

Figure 6: Pareto ranking

## 5.2 Pareto-based approaches

In the absence of information concerning the relative importance of the objectives, an individual can only be said to perform better than another if it dominates it. Therefore, non-dominated individuals should be assigned the same cost [7], e.g., zero. Deciding about the cost of dominated individuals is a more subjective matter. One alternative consists of assigning individuals a cost proportional to how many other individuals in the population dominate them (Figure 6), which also guarantees that non-dominated individuals are treated as desired. This is essentially the Pareto-ranking scheme proposed in [21].

Another popular Pareto-ranking scheme [7], also known as non-dominated sorting [22], consists of removing the non-dominated individuals (still ranked zero, for ease of comparison) from contention, finding the non-dominated individuals in the remaining population and assigning them rank 1, and so forth, until the whole population is ranked.

Both approaches guarantee that non-dominated individuals are all ranked best, and that all individuals are assigned better ranks than those individuals they dominate. However, the first ranking scheme does appear to be easier to interpret and analyse mathematically [23].

## 5.3 Incorporating preference information

When goal and/or priority information is available for the objectives, it may become possible to discriminate between some non-dominated solutions. For example, if degradation in objective components which meet their goals does not go beyond the goal boundaries, and results in the improvement of objective components which do not yet satisfy the corresponding goals, then it should be accepted. Similarly, in a dual priority setup [23], it is only important to improve on high priority objectives (i.e., constraints) until the corresponding goals are met, after which improvement should be sought for the remaining objectives. These considerations have been formalised in terms of a transitive relational operator (*preferability*), based on Pareto-dominance, but which selectively excludes objectives according to their priority and to whether or not they meet their goals.

For simplicity, only one level of priority will be considered here. The full, multiple priority version of the preferability operator is described in detail in [23]. Consider two objective vectors **u** and **v** and a goal vector **g**. Also, let the smile $\overset{u}{\smile}$ and the frown $\overset{u}{\frown}$ denote the components of **u** which meet their goals and those which do not, respectively. Assuming minimisation, one can write

$$\mathbf{u}^{\overset{u}{\smile}} \leq \mathbf{g}^{\overset{u}{\smile}} \quad \wedge \quad \mathbf{u}^{\overset{u}{\frown}} > \mathbf{g}^{\overset{u}{\frown}},$$

where the inequalities apply componentwise. This is equivalent to

$$\forall i \in \overset{\mathbf{u}}{\smile},\ u_i \leq g_i \quad \wedge \quad \forall i \in \overset{\mathbf{u}}{\frown},\ u_i > g_i$$

where $u_i$ and $g_i$ represent the components of **u** and **g**, respectively. Then, **u** is said to be preferable to **v** given **g** if and only if

$$(\mathbf{u}^{\overset{u}{\frown}} \mathrm{p}\!< \mathbf{v}^{\overset{u}{\frown}}) \vee \left\{ (\mathbf{u}^{\overset{u}{\frown}} = \mathbf{v}^{\overset{u}{\frown}}) \wedge \left[ (\mathbf{v}^{\overset{u}{\smile}} \not\leq \mathbf{g}^{\overset{u}{\smile}}) \vee (\mathbf{u}^{\overset{u}{\smile}} \mathrm{p}\!< \mathbf{v}^{\overset{u}{\smile}}) \right] \right\}$$

where **a** p< **b** denotes **a** dominates **b**. In other words, **u** will be preferable to **v** if and only if one of the following is true:

1. The violating components of **u** dominate the corresponding components of **v**.

2. The violating components of **u** are equal to the corresponding components of **v**, but **v** violates at least another goal.

3. The violating components of **u** are equal to the corresponding components of **v**, but **u** dominates **v** as a whole.

Like Pareto-dominance, this relation can be used to rank the individuals in a population by one of the methods described above.

## 6 Concluding remarks

In this paper, evolutionary optimisation was introduced, and an artificial model of evolution was given which encompasses most established EA variants. That model was then reinterpreted so as to accommodate multiple criteria optimisation problems. In the same light, it was shown how existing preferences may be combined with the notion of Pareto dominance by defining an alternative relation.

Much more could be said about evolutionary multi-criterion optimisation. Modern evolutionary multi-criterion optimisers have introduced additional mechanisms in the evolutionary process, including niche induction techniques, for maintaining diversity, and solution archiving, for preserving good solutions in the population. As a result, it has become increasingly less clear which algorithm works best in general, and increasing attention is being paid to experimental methodology for studying the performance of multiobjective optimisers.

In the mean time, existing evolutionary multi-criterion optimisers have been used in a wide range of industrial applications, making this one of the most promising research areas in evolutionary computing. The reader is referred to [24] for a comprehensive text book on Evolutionary Multiobjective Optimisation, and to [25, 26] for recent developments.

## References

[1] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*. MIT Press, 1990.

[2] P. Taylor, *Practical Foundations of Mathematics*. Cambridge University Press, 1999.

[3] I. Rechenberg, *Evolutionsstrategie, Optimierung technischer Systeme nach Prinzipen der biologischen Evolution*. Stuttgart: frommann-holzboog, 1973. In German.

[4] T. Bäck, F. Hoffmeister, and H.-P. Schwefel, "A survey of evolution strategies," in *Genetic Algorithms: Proceedings of the Fourth International Conference* (R. K. Belew and L. B. Booker, eds.), pp. 2–9, San Mateo, California: Morgan Kaufmann, 1991.

[5] D. B. Fogel, *System Identification Through Simulated Evolution: A machine learning approach to modelling*. Needham, Massachusetts: Ginn Press, 1991.

[6] J. H. Holland, *Adaptation in Natural and Artificial Systems*. Ann Arbor: The University of Michigan Press, 1975.

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

[8] D. B. Fogel, "Principles of evolutionary processes," in *Evolutionary Computation 1: Basic Algorithms and Operators* (T. Baeck, D. Fogel, and T. Michalewicz, eds.), ch. 4, pp. 23–26, Institute of Physics Publishing, 2000.

[9] D. E. Goldberg and J. Richardson, "Genetic algorithms with sharing for multimodal function optimization," in Grefenstette [27], pp. 41–49.

[10] K. Deb and D. E. Goldberg, "An investigation of niche and species formation in genetic function optimization," in Schaffer [28], pp. 42–50.

[11] G. Ochoa, I. Harvey, and H. Buxton, "Error thresholds and their relation to optimal mutation rates," in *European Conference on Artificial Life*, pp. 54–63, 1999.

[12] R. K. Belew, "Evolution, learning and culture: Computational metaphors for adaptive algorithms," *Complex Systems*, vol. 4, pp. 11–49, 1990.

[13] J. E. Baker, "Adaptive selection methods for genetic algorithms," in Grefenstette [29], pp. 101–111.

[14] J. E. Baker, "Reducing bias and inefficiency in the selection algorithm," in Grefenstette [27], pp. 14–21.

[15] L. Booker, "Improving search in genetic algorithms," in *Genetic Algorithms and Simulated Annealing* (L. Davis, ed.), Research Notes in Artificial Intelligence, ch. 5, pp. 61–73, London: Pitman, 1987.

[16] G. Syswerda, "Uniform crossover in genetic algorithms," in Schaffer [28], pp. 2–9.

[17] R. A. Caruana, L. J. Eshelman, and J. D. Schaffer, "Representation and hidden bias II: Eliminating defining length bias in genetic search via shuffle crossover," in *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence* (N. S. Sridharan, ed.), pp. 750–755, Morgan Kaufmann, 1989.

[18] A. Ben-Tal, "Characterization of Pareto and lexicographic optimal solutions," in *Multiple Criteria Decision Making Theory and Application* (G. Fandel and T. Gal, eds.), vol. 177 of *Lecture Notes in Economics and Mathematical Systems*, pp. 1–11, Berlin: Springer-Verlag, 1980.

[19] C. M. Fonseca and P. J. Fleming, "An overview of evolutionary algorithms in multiobjective optimization," *Evolutionary Computation*, vol. 3, Spring 1995.

[20] J. D. Schaffer, "Multiple objective optimization with vector evaluated genetic algorithms," in Grefenstette [29], pp. 93–100.

[21] C. M. Fonseca and P. J. Fleming, "Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization," in *Genetic Algorithms: Proceedings of the Fifth International Conference* (S. Forrest, ed.), pp. 416–423, San Mateo, CA: Morgan Kaufmann, 1993.

[22] N. Srinivas and K. Deb, "Multiobjective optimization using nondominated sorting in genetic algorithms," *Evolutionary Computation*, vol. 2, Fall 1994. To appear.

[23] C. M. Fonseca and P. J. Fleming, "Multiobjective optimization and multiple constraint handling with evolutionary algorithms I: A unified formulation," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 28, no. 1, pp. 26–37, 1995.

[24] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley & Sons, 2001.

[25] E. Zitzler, K. Deb, L. Thiele, C. A. C. Coello, and D. Corne, eds., *Evolutionary Multi-Criterion Optimization, First International Conference*, vol. 1993 of *Lecture Notes in Computer Science*. Springer, 2001.

[26] C. M. Fonseca, P. J. Fleming, E. Zitzler, K. Deb, and L. Thiele, eds., *Evolutionary Multi-Criterion Optimization, Second International Conference*, vol. 2632 of *Lecture Notes in Computer Science*. Springer, 2003.

[27] J. J. Grefenstette, ed., *Genetic Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic Algorithms*, Lawrence Erlbaum, 1987.

[28] J. D. Schaffer, ed., *Proceedings of the Third International Conference on Genetic Algorithms*. San Mateo, CA: Morgan Kaufmann, 1989.

[29] J. J. Grefenstette, ed., *Genetic Algorithms and Their Applications: Proceedings of the First International Conference on Genetic Algorithms*, Lawrence Erlbaum, 1985.