12

Multiobjective Evolutionary Algorithms

In this chapter we describe the application of evolutionary techniques to a particular class of problems, namely multiobjective optimisation. We begin by introducing this class of problems and the particularly important notion of Pareto optimality. We then look at some of the current state-of-the-art multiobjective EAs (MOEAs) for this class of problems and examine the ways in which they make use of concepts of different evolutionary spaces and techniques for promoting and preserving diversity within the population.

12.1 Multiobjective Optimisation Problems

In the majority of our discussions in previous chapters we have made free use of analogies such as adaptive landscapes under the assumption that the goal of the EA in an optimisation problem is to find a single solution that maximises a fitness value that is directly related to a single underlying measure of quality. We also discussed a number of modifications to EAs that are aimed at preserving diversity so that a *set* of solutions is maintained; these represent niches of high fitness, but we have still maintained the conceptual link to an adaptive landscape defined via the assignment of a *single* quality metric (objective) to each of the set of possible solutions.

We now turn our attention to a class of problems that are currently receiving a lot of interest within the optimisation community, and in practical applications. These are the so-called **multiobjective problems** (MOPs), where the quality of a solution is defined by its performance in relation to several, possibly conflicting, objectives. In practice it turns out that a great many applications that have traditionally been tackled by defining a single objective function (quality function) have at their heart a multiobjective problem that has been transformed into a single-objective function in order to make optimisation tractable.

To give a simple illustration (inspired by [334]), imagine that we have moved to a new city and are in the process of looking for a house to buy. There are

A.E. Eiben, J.E. Smith, *Introduction to Evolutionary Computing*,
Natural Computing Series, DOI 10.1007/978-3-662-44874-8_12

a number of factors that we will probably wish to take into account, such as: number of rooms, style of architecture, commuting distance to work and method, provision of local amenities, access to pleasant countryside, and of course, price. Many of these factors work against each other (particularly price), and so the final decision will almost inevitably involve a compromise, based on trading off the house's rating on different factors.

The example we have just presented is a particularly subjective one, with some factors that are hard to quantify numerically. It does exhibit a feature that is common to multiobjective problems, namely that it is desirable to present the user with a diverse set of possible solutions, representing a range of different trade-offs between objectives.

The alternative is to assign a numerical quality function to each objective, and then combine these scores into a single fitness score using some (usually fixed) weighting. This approach, often called **scalarisation**, has been used for many years within the operations research and heuristic optimisation communities (see [86, 110] for good reviews), but suffers from a number of drawbacks:

- the use of a weighting function implicitly assumes that we can capture all of the user's preferences, even before we know what range of possible solutions exist.
- for applications where we are repeatedly solving different instances of the same problem, the use of a weighting function assumes that the user's preferences remain static, unless we explicitly seek a new weighting every time.

For these reasons optimisation methods that simultaneously find a *diverse* set of high-quality solutions are attracting increasing interest.

12.2 Dominance and Pareto Optimality

The concept of **dominance** is a simple one: given two solutions, both of which have scores according to some set of objective values (which, without loss of generality, we will assume to be maximised), one solution is said to dominate the other if its score is at least as high for all objectives, and is strictly higher for at least one. We can represent the scores that a solution A gets for n objectives as an n-dimensional vector $\overline{a}$. Using the $\succeq$ symbol to indicate domination, we can define $A \succeq B$ formally as:

$$A \succeq B \iff \forall i \in \{1, \ldots, n\} \;\; a_i \geq b_i, \text{ and } \exists i \in \{1, \ldots, n\}, \; a_i > b_i.$$

For conflicting objectives, there exists no single solution that dominates all others, and we will call a solution **nondominated** if it is not dominated by any other. All nondominated solutions possess the attribute that their quality cannot be increased with respect to any of the objective functions without

detrimentally affecting one of the others. In the presence of constraints, such solutions usually lie on the edge of the feasible regions of the search space. The set of all nondominated solutions is called the **Pareto set** or the Pareto front.

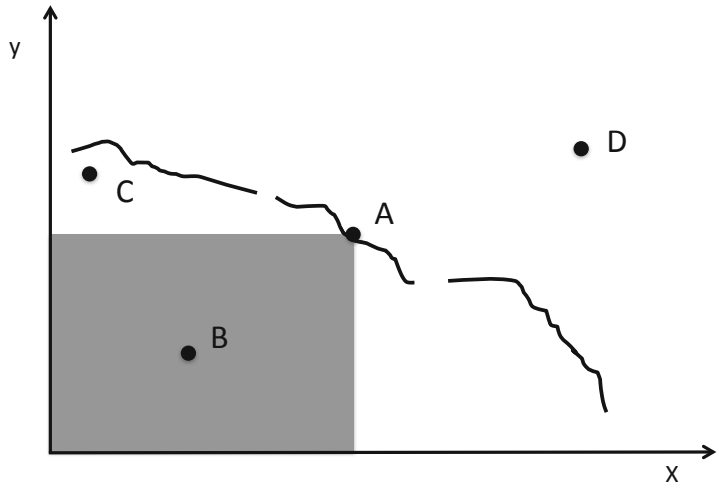

Fig. 12.1. Illustration of the Pareto front. The x- and y-axes represent two conflicting objectives subject to constraints. The quality of solutions is represented by their x and y values (larger is better). Point A dominates B and all other points in the grey area. A and C do not dominate each other. The line represents the Pareto set, of which point A is an example. Solutions above and to the right of the line, such as D, are infeasible

In Figure 12.1 this front is illustrated for two conflicting objectives that are both to be maximised. This figure also illustrates some of the features, such as nonconvexity and discontinuities, frequently observed in real applications that can cause particular problems for traditional optimisation techniques using often sophisticated variants of scalarisation to identify the Pareto set. EAs have a proven ability to identify high-quality solutions in high-dimensional search spaces containing difficult features such as discontinuities and multiple constraints. When coupled with their population-based nature and their ability for finding and preserving diverse sets of good solutions, it is not surprising that EA-based methods are currently the state of the art in many multiobjective optimisation problems.

12.3 EA Approaches to Multiobjective Optimisation

There have been many approaches to multiobjective optimisation using EAs, beginning with Schaffer's vector-evaluated genetic algorithm (VEGA) in 1984 [364]. In this algorithm the population was randomly divided into subpopulations that were then each assigned a fitness (and subject to selection) according to a different objective function, but parent selection and recombination were performed globally. This modification was shown to be enough to preserve an approximation to the Pareto front for a few generations, but not indefinitely.

Subsequent to this, Goldberg suggested the use of fitness based on dominance rather than on absolute objective scores [189], coupled with niching and/or speciation methods to preserve diversity, and this breakthrough triggered a dramatic increase in research activity in this area. We briefly describe some of the best-known algorithms below, noting that the choice of representation, and hence variation operators, are entirely problem dependent, and so we concentrate on the way that fitness assignment and selection are performed.

12.3.1 Nonelitist Approaches

Amongst the first algorithms to explicitly exert selection pressure towards the discovery of nondominated solutions are discussed below:

Fonseca and Fleming's multiobjective genetic algorithm (MOGA) [175] assigns a raw fitness to each solution equal to the number of members of the current population that it dominates, plus one. It uses fitness sharing amongst solutions of the same rank, coupled with fitness-proportionate selection to help promote diversity.

Srinivas and Deb's nondominated sorting genetic algorithm (NSGA) [417] works in a similar way, but assigns fitness based on dividing the population into a number of fronts of equal domination. To achieve this, the algorithm iteratively seeks all the nondominated points in the population that have not been labelled as belonging to a previous front. It then labels the new set as belonging to the current front, and increments the front count, repeating until all solutions have been labelled. Each point in a given front gets as its raw fitness the count of all solutions in inferior fronts. Again fitness sharing is implemented to promote diversity, but this time it is calculated considering only members from that individual's front.

Horn et al.'s niched Pareto genetic algorithm (NPGA) [223] differs in that it uses a modified version of tournament selection rather than fitness proportionate with sharing. The tournament operator works by comparing two solutions first on the basis of whether they dominate each other, and then second on the number of similar solutions already in the new population.

Although all three of these algorithms show good performance on a number of test problems, they share two common features. The first of these is that

the performance they achieve is heavily dependent on a suitable choice of parameters in the sharing/niching procedures. The second is that they can potentially lose good solutions.

12.3.2 Elitist Approaches

During the 1990s much work was done elsewhere in the EA research community, developing methods for reducing dependence on parameter settings (Chaps. 7 and 8). Theoretical breakthroughs were achieved showing that single-objective EAs converge to the global optimum on some problems, providing that an elitist strategy (Sect. 5.3.2) is used. In the light of this research Deb and coworkers proposed the revised NSGA-II [112], which still uses the idea of non-dominated fronts, but incorporates the following changes:

- A crowding distance metric is defined for each point as the average side length of the cuboid defined by its nearest neighbours in the same front. The larger this value, the fewer solutions reside in the vicinity of the point.
- A $(\mu + \lambda)$ survivor selection strategy is used (with $\mu = \lambda$). The two populations are merged and fronts assigned. The new population is obtained by accepting individuals from progressively inferior fronts until it is full. If not all of the individuals in the last front considered can be accepted, they are chosen on the basis of their crowding distance.
- Parent selection uses a modified tournament operator that considers first dominance rank then crowding distance.

As can be seen, this achieves elitism (via the plus strategy) and an explicit diversity maintenance scheme, as well as reduced dependence on parameters.

Two other prominent algorithms, the strength Pareto evolutionary algorithm (SPEA-2) [475] and the Pareto archived evolutionary strategy (PAES) [251], both achieve the elitist effect in a slightly different way by using an archive containing a fixed number of nondominated points discovered during the search process. Both maintain a fixed sized archive, and consider the number of archived points close to a new solution, as well as dominance information, when updating the archive.

12.3.3 Diversity Maintenance in MOEAs

To finish our discussion on MOEAs it is appropriate to consider how sets of diverse solutions can be maintained during evolution. It should be clear from the descriptions of the MOEAs above that all of them use explicit methods to enforce preservation of diversity, rather than relying simply on implicit measures such as parallelism (in one form or another) or artificial speciation.

In single-objective optimisation, explicit diversity maintenance methods are often combined with implicit speciation methods to permit the search for optimal solutions within the preserved niches. The outcome of this is a *few*

highly fit diverse solutions, often with multiple copies of each (Fig. 5.4). In contrast to this, the aim of MOEAs is to attempt to distribute the population *evenly* along the current approximation to the Pareto front. This partially explains why speciation techniques have not been used in conjunction with the explicit measures. Finally, it is worth noting that the more modern algorithms discussed have abandoned fitness sharing in favour of direct measures of the distance to the nearest nondominating solution, more akin to crowding.

12.3.4 Decomposition-Based Approaches

An unavoidable problem of trying to evenly represent an approximation of the Pareto front is that this approach does not scale well as the dimensionality of the solution space increases above 5–10 objectives. One recent method that has gathered a lot of attention is the decomposition approach taken by Zhang's MOEA-D algorithm [474] which shares features from the single-objective weighted sum approach and the population-based approaches. Rather than use a single weighted combination, MOEA-D starts by evenly distributing a set of N weight vectors in the objective space, and for each building a list of its T closest neighbours (measured by Euclidean distances between them). It then creates and evolves a population of N individuals, each associated with one of the weight vectors, and uses it to calculate a single fitness value. What differentiates this from simply being N parallel independent searches is the use of the neighbourhood sets to structure the population, with selection and recombination only happening within demes, as in cellular EAs (Sect. 5.5.7). Periodically the location of the weight vectors and the neighbour sets are recalculated so that the focus of the N parallel searches reflects what has been discovered about the solution space. MOEA-D, and variants on the original algorithm, have been shown to perform equivalently to dominance-based methods in low dimensions, and to scale far better to many-objective problems with 5 or more conflicting dimensions.

12.4 Example Application: Distributed Coevolution of Job Shop Schedules

An interesting application, which makes use of many of the ideas in this chapter, and also some in Chap. 15, can be seen in Husbands' distributed coevolutionary approach to multiobjective problems [225]. In this approach he uses a coevolutionary model to tackle a complex multiobjective, multi-constraint problem, namely a generalised version of job shop scheduling. Here a number of items need to be manufactured, each requiring a number of operations on different machines. Each item may need a different number of operations, and in general the order of the operations may be varied, so that the problem of finding an optimal production plan for *one* item is itself NP-hard. The usual approach to the multiple task problem is to optimise each plan

individually, and then use a heuristic scheduler to interleave the plans so as to obtain an overall schedule. However, this approach is inherently flawed because it optimises the plans in isolation rather than taking into consideration the availability of machines, etc.

Husbands' approach is different: he uses a separate population to evolve plans for each item and optimises these concurrently. In this sense we have a MOP, although the desired final output is a single set of plans (one for each item) rather than a set of diverse schedules. A candidate plan for one item gets evaluated in the context of a member from each other population, i.e., the fitness value (related to time and machining costs) is for a complete production schedule. An additional population is used to evolve 'arbitrators', which resolve conflicts during the production of the complete schedule.

Early experiments experienced problems with premature loss of diversity. These problems are treated by the use of an implicit approach to diversity preservation, namely the use of a diffusion model EA. Furthermore, by colocating one individual from each population in each grid location, the problem of partner selection (Sect. 15.2.1) is neatly solved: a complete solution for evaluation corresponds to a grid cell.

We will not give details of Husbands' representation and variation operators here, as these are highly problem specific. Rather we will focus on the details of his algorithm that were aimed at aiding the search for high-class solutions. The first of these is, of course, the use of a coevolutionary approach. If a single population were used, with a solution representing the plans for all items, there would be a greater likelihood of genetic hitchhiking (see Sect. 16.1 for a description), whereby a good plan for one item in the initial population would take over, even if the plans for the other items were poor. By contrast, the decomposition into different subpopulations means that the good plan can at worst take over one population.

The second feature that aids the search over diverse local optima is the use of a diffusion model approach. The implementation uses a 15-by-15 square toroidal grid, thus a population size of 225. Plans for 5 items, each needing between 20 and 60 operations, were evolved, so in total there were 6 populations, and each cell contained a plan for each of the 5 items plus an arbitrator. A generational approach is used: within each generation each cell's populations are 'bred', with a random permutation to decide the order in which cells are considered.

The breeding process within each cell is iterated for each population and consists of the following steps:

1. Generate a set of points to act as neighbours by iteratively generating random lateral and vertical offsets from the current position. A binomial approximation to a Gaussian distribution is used, which falls off sharply for distances more than 2 and is truncated to distance 4.
2. Rank the cells in this neighbourhood according to cost, and select one using linear ranking with $s = 2$.

3. Take the member of the current population from the selected cell and the member in the current cell, and generate an offspring via recombination and mutation.
4. Choose a cell from the neighbourhood using inverse linear ranking.
5. Replace the member of the current population in that cell with the newly created offspring.
6. Re-evaluate all the individuals in that cell using the newly created offspring.

The results presented from this technique showed that the system managed to evolve low-cost plans for each item, together with a low total schedule time. Notably, even after several thousand iterations, the system had still preserved a number of diverse solutions.

For exercises and recommended reading for this chapter, please visit www.evolutionarycomputation.org.