Evolutionary population dynamics and multi-objective optimisation problems

Lewis, Andrew; Mostaghim, Sanaz; Randall, Marcus

Published in:
Multi-Objective Optimization in Computational Intelligence: Theory and Practice

DOI:
10.4018/978-1-59904-498-9.ch007

Evolutionary Population Dynamics and Multi-Objective Optimisation Problems

Andrew Lewis
Institute for Integrated and Intelligent Systems
Griffith University
Brisbane, Queensland
Australia
Ph: +61 7 373 56789
Fax: +61 7 373 56650
a.lewis@griffith.edu.au

Sanaz Mostaghim
Institute AFIB
University of Karlsruhe
Germany
Ph: +49 721 608 6554
Fax: +49 721 693717
smo@aifb.uni-karlsruhe.de

Marcus Randall
School of Information Technology
Bond University
Gold Coast, Queensland
Australia
Ph: +61 7 55953361
Fax: +61 7 55953320
mrandall@bond.edu.au

This chapter appears in *Multi-Objective Optimization In Computational Intelligence: Theory And Practice* edited by Bui and Alam.

Evolutionary Population Dynamics and Multi-Objective Optimisation Problems

Abstract

Problems for which many objective functions are to be simultaneously optimised are widely encountered in science and industry. These multiobjective problems have also been the subject of intensive investigation and development recently for metaheuristic search algorithms such as ant colony optimisation, particle swarm optimisation and extremal optimisation. In this chapter, a unifying framework called evolutionary programming dynamics (EPD) is examined. Using underlying concepts of self organised criticality and evolutionary programming, it can be applied to many optimisation algorithms as a controlling metaheuristic, to improve performance and results. We show this to be effective for both continuous and combinatorial problems.

Keywords: Algorithms, Heuristics, Optimization Methods.

INTRODUCTION

Due to the large number of applications in science and industry, Multiobjective Optimisation using Evolutionary Algorithms (MOEAs) have been increasingly studied during the last decade (Coello Coello, Van Veldhuizen & Lamont, 2002; Deb, 2001; Zitzler, 1999). There are many issues to be resolved for effective use of MOEAs, such as developing new algorithms to obtain solutions with good diversity and convergence, designing metrics for measuring the quality of the achieved solutions, producing test functions in static and dynamic environments etc. Any new development in these areas is valuable for scientific and industrial applications. However, solving large scale problems with a large number of objectives is still a major challenge (Purshouse & Flemming, 2003; Deb, 2001).

In this chapter, we outline the development and use of evolutionary population dynamics (EPD) as a metaheuristic for population-based optimisation algorithms. These include, but are not limited to, ant colony optimisation (ACO) (Dorigo, 1999), Extremal Optimisation (EO) (Boettcher & Percus, 2000) and Particle Swarm Optimisation (PSO) (Eberhart & Kennedy, 1995). This approach can be applied to both continuous and combinatorial problems for single-valued and multiobjective problems.

MULTIOBJECTIVE ORIENTED METAHEURISTICS

As preliminary background, we describe three well-known metaheuristics: particle swarm optimisation, ant colony optimisation and extremal optimisation. The general mechanics of each method is briefly outlined along with how they have been applied to multiobjective optimisation.

2.1 Particle Swarm Optimisation

PSO is motivated from the simulation of social behaviour of animals (Eberhart & Kennedy, 1995; Kennedy & Eberhart, 1995; Englebrecht, 2005). PSO is a population based technique, similar in some respects to evolutionary algorithms, except that potential solutions (particles) move, rather than evolve, through the search space. The rules, or particle dynamics, which govern this movement, are inspired by models of swarming and flocking. Each particle has a position and a velocity, and experiences linear spring-like attractions towards two guides:

1. The best position attained by that particle so far (local guide), and
2. The best position found by the swarm as a whole (global guide),

where “best” is in relation to evaluation of an objective function at that position. The global guide therefore enables information sharing between particles, whilst the local guides serve as individual particle memories.

The optimisation process is iterative. At each iteration, the acceleration vectors of all the particles are calculated based on the positions of the corresponding guides. Then, this acceleration is added to the velocity vector, the updated velocity is constricted so that the particles progressively slow down, and this new velocity is used to move the individual from the current to the new position.

Due to the success of particle swarm optimisation in single objective optimisation, in recent years more and more attempts have been made to extend PSO to the domain of multiobjective problems, see for example (Mostaghim & Teich, 2003; Alvarez-Benitez, Everson & Fieldsend, 2005; Parsopoulos & Vrahatis, 2002; Mostaghim, 2005). The main challenge in multiobjective particle swarm optimisation (MOPSO) is to select the global and local guides such that the swarm is guided towards the Pareto optimal front and maintains sufficient diversity. In MOPSO, the set of non-dominated solutions must be used to determine the global guide for each particle. Selecting, or constructing, the guide from this set for each particle of the population is a very difficult yet important problem for attaining convergence and diversity of solutions. Several methodologies in MOPSO for selecting the global guide and their influences on the convergence and diversity of solutions are being explored (Fieldsend & Singh, 2002; Reyes-Sierra & Coello Coello, 2006; Mostaghim, 2005; Alvarez-Benitez et al., 2005; Ireland, Lewis, Mostaghim & Lu, 2006).

In the early stage of developing MOPSO algorithms, Parsopulos and Vrahatis (2002) modified the idea of VEGA - Vector Evaluated Genetic Algorithm - to MOPSO. Their algorithm was based on an aggregated objective function, changing the multiobjective problem to that of a single objective. Hu and Eberhart (2002) proposed a method in which only one objective was optimised at a time. Their method is efficient for problems with a low number of objectives and problems which are not sensitive to the order of objectives. Recently, most research in the area of MOPSO has concentrated on the selection of the global guide for each individual. Mostaghim and Teich (2003) introduced the Sigma method for guiding particles towards the Pareto-front, Ireland et al. (2006) have suggested an artificial, constructed guide, and Fieldsend and Singh (2002) use an elite archive. Coello Coello and Lechuga (2002) suggest using a random selection and a repository for saving the non-dominated solutions. Alvarez-Benitez et al. (2005) have introduced a selection schema for solutions that dominate many particles for use as global guides. In recent work, the task of selecting the local guide has been demonstrated by Branke and Mostaghim (2006) to be as important as selecting the global guide.

Practical application of MOPSO is in its early stages. There are a number of hybrid-MOPSO algorithms for solving real-world problems, for example Mostaghim and Halter (2006), and Parallel MOPSO has been proposed recently by Mostaghim, Branke and Schmeck (2006). For a more extensive treatment of MOPSO and PSO techniques in general, the reader is referred to Reyes-Sierra and Coello Coello (2006).

2.2 Ant Colony Optimisation

Ant Colony Optimisation (ACO) (Dorigo, 1999) is a population optimisation paradigm encompassing a range of metaheuristics based on the evolutionary mechanics of natural ant colonies. These techniques have been applied extensively to benchmark problems such as the travelling salesman problem (TSP), the job sequencing problem and the quadratic assignment problem (QAP). Work on more complex problems, that have difficult constraints in such areas as transportation and telecommunications, has also been undertaken (Dorigo, 1999). Like other evolutionary algorithms, populations of solutions evolve over time. The major difference is that ACO represents a set of constructive techniques, i.e., each ant at each step of the generalised algorithm adds a component (such as the next city for the TSP) to its solution. Using simulated chemical or pheromone markers as a collective form of self adaptation, populations produce increasingly better solutions.

In terms of multiobjective optimisation, an illustrative sample of work will be surveyed here. A good overview of this topic is found in Garcia-Martinez, Cordon and Herrera (2007). In the main, ACO has been used to solve specific instances of multiobjective problems. These problems are from diverse areas, ranging from multi-dimensional TSPs and vehicle routing problems to sharemarket portfolio selection and planning water irrigation channels. While they have been tailored to these problems, the main theme has been the use of different colonies to optimise each objective function.

Garcia-Martinez, Cordon and Herrera (2004, 2007) compare a number of published multiobjective ant colony optimisation (MOACO) methods on a single set of bi-criteria TSPs. These methods are from Mariano and Morales (1999), Iredi, Merkle and Middendorf (2001), Gambardella, Taillard and Agazzi (1999) and Doerner, Gutjahr, Hartl and Strauss (2004). For the first of these (named MOAQ), families of ants are used to optimise each objective. Each ant of the family learns from its same number in the preceding family. Additionally, infeasible solutions are penalised. Iredi et al. (2001) propose two methods they call BicriterionAnt and BicriterionMC. Both use different pheromone repositories for the two objective functions. The main difference is that the former allows its ants to update both repositories while the latter will update one or the other (at each step of an iteration). Doerner et al. (2004) use a variation on the idea of separate colonies in a scheme called COMPETants. After each iteration of the algorithm, the better performing colony gets a greater number of ants for the next generation. In contrast, the multiple ant colony system of Gambardella et al. (1999) uses a single pheromone matrix for the each colony, but uses multiple visibility heuristics.

The results of Garcia-Martinez et al. (2004) indicate the MOACOs could return better sets of solutions (in terms of the Pareto-front) than other more specialised algorithms (specifically NSGA-II and SPEA2). Apart from COMPETants and MOAQ, the MOACOs performed equally well.

Population ACO (PACO)[i] has also been used to solve multiple objective optimisation problems (Guntsch & Middendorf, 2003). Rather than keeping separate colonies of ants for each objective function, PACO creates different sub-populations to update each of the pheromone matrices (one for each objective). A variation of this is Crowding PACO (CPACO) (Angus, 2006) in which a population, without sub-populations, is maintained. Each solution of the current population of ants is compared (in terms of solution quality) to a subset of the population. Given that two solutions are approximately similar, the old population member will be replaced if the other dominates it. Using a set of bi-criteria TSPs, it was shown that that CPACO was able to outperform PACO.

2.3 Extremal Optimisation

EO is a relatively new metaphor for solving functional and combinatorial optimisation problems. It is based on the Bak-Sneppen model for self organising criticality (Bak & Tang, 1987; Bak & Sneppen, 1993). To date only a limited number of combinatorial problems have been solved with this method. As it is relatively new in the field, compared to other techniques such as ACO, genetic algorithms (Goldberg, 1989) and PSO there exists wide scope to apply and extend this heuristic.

Boettcher and Percus (1999, 2000, 2003) describe the general tenets of EO. Nature can be seen as an optimising system in which the aim is to allow competitive species to populate environments. In the course of the evolutionary process, successful species will have the ability to adapt to changing conditions while the less successful will suffer and may become extinct. This notion extends to the genetic level as well. Poorly performing genes are replaced using random mutation. Over time, if the species does not become extinct, its overall fitness will increase.

As EO is an emerging metaheuristic, relatively little has been done on it in connection with multiobjective optimisation. Galski, de Sousa, Ramos and Muraoka (2007) present an EO algorithm to find the optimal design of a simplified configuration of a thermal control system for a spacecraft platform. The objectives are to minimise the difference between target and actual temperatures on radiation panels as well as to minimise battery heater power dissipation. Using EO, two possible solutions were found. As the authors combined both objectives into a single function, the designs were really only able to satisfy the first objective. Future designs will optimise both objectives using a Pareto-front approach.

EVOLUTIONARY POPULATION DYNAMICS

The optimisation process is, at its core, a process of gradual improvement. Some trial solution to a problem is proposed, measured against some objective scale, and algorithmic means applied to transform the solution to some ideal solution. It is a simple, but insightful, observation that the trial solution evolves. Early optimisation methods drew on a foundation of mathematical and geometric methods to search for particular solutions to systems of equations: the classical methods of gradient descent and direct search. In recent years practitioners have sought inspiration from nature, giving rise to such methods as genetic algorithms, evolutionary programming methods, swarm intelligence, ant colony algorithms and immune system simulation. Some of the most powerful heuristics are modelled on direct observation of the processes of evolution of species. Populations of solutions are manipulated to mimic evolutionary adaptation, in which species gradually become more suited to their environment or, to put it in optimisation terms, alter their characteristics (parameters) to become better adapted to the environmental pressures (objectives) imposed upon them.
The evolutionary programming methods of Fogel (1962) are a simple, robust and highly parallel approach to implementing these concepts. This class of methods in evolutionary computation apply a random mutation to each member of a population, generating a single offspring. However, unlike several other approaches, no recombination operators are applied. Some form of selection takes place, and half the combined population of parents and offspring enter the next generation. It is important to note that this is a phylogenetically-oriented approach: population members are considered as representative of species, not individual members within those species; hence the lack of recombination operators which mimic sexual reproduction.

The theory of self-organised criticality can give additional insight into emergent complexity in nature. Bak (1996) contends that the critical state is “the most efficient state that can actually be reached dynamically”. Understanding “efficient” to mean “well-adapted”, this holds promise of another means to improve population fitness. In this state, a population in an apparent equilibrium evolves episodically in spurts, a phenomenon known as punctuated

equilibrium. Local change may affect any other element in the system, and this delicate balance arises without any external, organising force.

Key to understanding the mechanics of the method is the observation that evolution progresses by selecting against the few most poorly adapted species rather than by expressly breeding those species best adapted to their environment. So the algorithms derived by reference to Bak's work do not have as their main aim the direct improvement of individual solutions, but the improvement of the population, as a whole, by removing poor solutions from it.

Using these concepts, an algorithm has been developed: Evolutionary Programming using Self Organising Criticality (EPSOC). This proved to be quite successful when applied to a range of real-world optimisation problems (Lewis, Abramson & Peachey, 2003). Since EPSOC's operation is basically to apply evolutionary population dynamics (EPD) to some population of solutions, and several contemporary multiobjective optimisation methods are based on manipulation of populations, potentially EPD could be applied as a controlling metaheuristic to improve their performance. In this work its application to ACO, EO and PSO is described, but it should be evident that it is equally applicable to a wide range of population-based algorithms.

We first describe the mechanics of EPSOC, and then how the integration with other optimisation algorithms is achieved.

3.1 EPSOC

EPSOC (Lewis et al., 2003) is a relatively new metaheuristic that follows the method of Fogel (1962) but with an additional selection operator from the Bak-Sneppen model. The steps of the algorithm are given in Algorithm 1.

Each set of values defining a trial solution is independent of all others. Therefore, the evaluation of trial solutions can be performed concurrently. Since the evaluation of the objective function generally dominates the execution time, extremely high parallel efficiency can be achieved.

EPSOC is a straightforward implementation of the nearest-neighbour, punctuated equilibrium model as an optimisation algorithm. It is novel in that, by considering the trial solution vectors as defining a location in an n-dimensional space, the spatial behaviour of the model is realised naturally. The algorithm has a high degree of greediness as it maintains a large elite (in EPSOC, half the total population). This can be viewed as a constructive operator. It encourages gradual improvement in the better half of the population. Over time, the mutated population members move into the protected half of the population. Thus the median of the population moves toward better objective function values.

Algorithm 1: The EPSOC algorithm

1: Initialise a random, uniformly-distributed population, and evaluate each trial solution
2: **for** a preset number of iterations **do**
3: Sort the population by objective function value
4: Select a set, *B*, of the worst members of the population. For each member of *B*, add to the set its two nearest neighbours in solution space that are not already members of the set, or from the best half of the sorted population
5: Reinitialise the solutions of the selected set, *B*. For all other members of the population, apply some (generally small) random, uniformly distributed mutation.
6: Evaluate each new trial solution.
7: For each of the trial solutions, if it has a better objective function value than its original, retain it.
8: **end for**
9: **end**

3.2 Applying EPD

The development of EPSOC allows us to investigate the use of standard metaheuristics (such as ACO, EO and PSO) as subservient heuristics to it. We aim to demonstrate that this hybrid approach may be used on a range of real-world problems.

A large component of this work is the determination of the allocation of sufficient parallel computing resources to the hybrid components.

To apply EPD it is, of course, necessary to have a subject population. For optimisation algorithms that are inherently population-based, such as ACO[ii] and PSO, EPD can be directly applied to their internal population. This is conceptually illustrated in Figure 1. The algorithmic power of the subservient heuristic acts on the population and gradually improves it. It can be expected that the median of the population will tend to move toward generally "better" solutions. Simultaneously, EPD acts to remove the least fit solutions. By this means the motion of the population median, as a measure of the overall population fitness, will be accelerated, leading to faster convergence rates.

However, if the method is *not* population-based, as with EO, a population on which EPD can act must be constructed. This is achieved by initiating a collection of several, independent searches using the sub-heuristic. The resulting "population" is subject to EPSOC's evolutionary population dynamics and, at each iteration, each member of the population carries out a metaheuristic algorithm search for some number of internal iterations to deliver an improved solution. This hybrid approach is effectively a *nested loop* of the two individual methods. For the example considered here, using EO, both components of the hybrid use a user-specified, maximum iteration count for their termination criterion. The hybrid algorithm is thus simple, robust and easy to control. The conceptual operation of the hybrid (using EO as the example subservient heuristic) is illustrated in Figure 2.

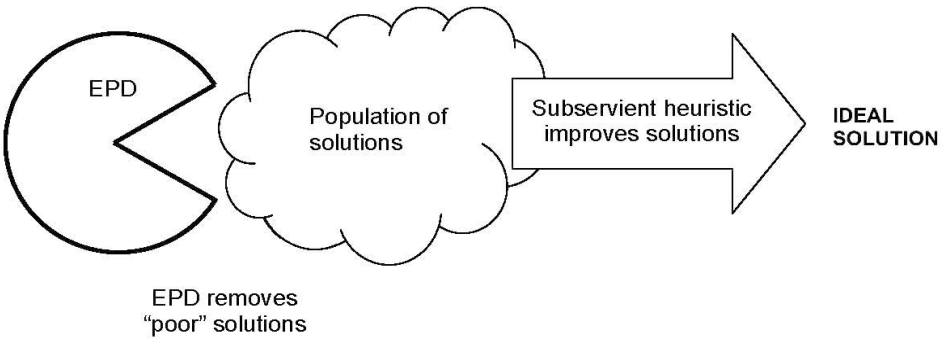


Figure 1: Functional structure of EPD applied to population-based methods

EPD has been used to good effect in combination with EO (Randall & Lewis, 2006), applied to a single-objective problem. To extend this approach for application to multiobjective problems presents a further complication. EPD assumes the population is ranked by fitness – the least fit solutions will be replaced. However, in multiobjective optimisation, as previously stated, there are no single best and worst solutions. Correctly selecting poor solutions for elimination has a great impact on the convergence of algorithms and quality of solutions obtained. Here, we examine several techniques for applying EPD to multiobjective problems and therefore develop a new methodology to solve large scale multiobjective optimisation problems.

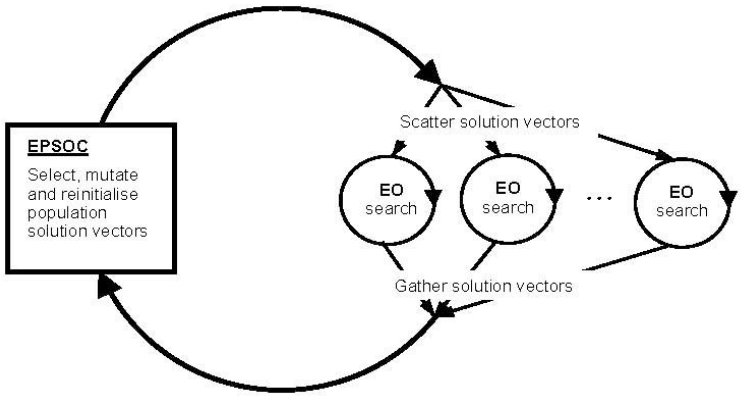


Figure 2: Functional structure of EPSOC and EO hybrid (© 2006 IEEE)

3.3 MOPSO and EPD

One possible combination of MOPSO and EPD is shown in Algorithm 2. In order to find the poor particles in the population, a hypervolume metric can be used. This metric can, in fact, be any quantitative measurement for convergence and diversity of solutions. Hypervolume is the volume between a particle and a reference point in the objective space (Figure 3). For finding the poor solutions, the total hypervolume of the population is calculated by $TotalHV$ = ComputeHyperVolume(P_t). Then the hypervolume of every single particle is computed as HV_i and being compared to a threshold value called TH. This value can be set by using the

TotalHV like one third of the total hypervolume. If a solution is categorised as poor that solution must be renewed (Step 3 in the Algorithm 2). The way a particle is renewed has a large impact on the solutions obtained. The two simple ways are:

1. find a random new position for the particle. This is like adding a turbulence factor to the poor solutions.
2. find a new position close to the particles in the Archive.

Algorithm 2: MOPSO+EPD Algorithm

---

1. **Initialisation**: Initialise population P_t, $t = 0$:
2. **Evaluation**: Evaluate(P_t)
3. **Renew Poor Solutions**:

TotalHV = ComputeHyperVolume(P_t)
for $i = 1$ to N **do**
  HV_i = ComputeHyperVolume $\left(\vec{x}_t^{\,i}\right)$
  if $HV_i < TH$ **then**
    Renew $\left(\vec{x}_t^{\,i}\right)$
  end if
end for

4. **Update**: A_{t+1} = Update(P_t, A_t)
5. **Move**: P_{t+1} = Move(P_t, A_t)
6. **Termination**: Unless a termination criterion is met $t = t + 1$ and goto Step 2

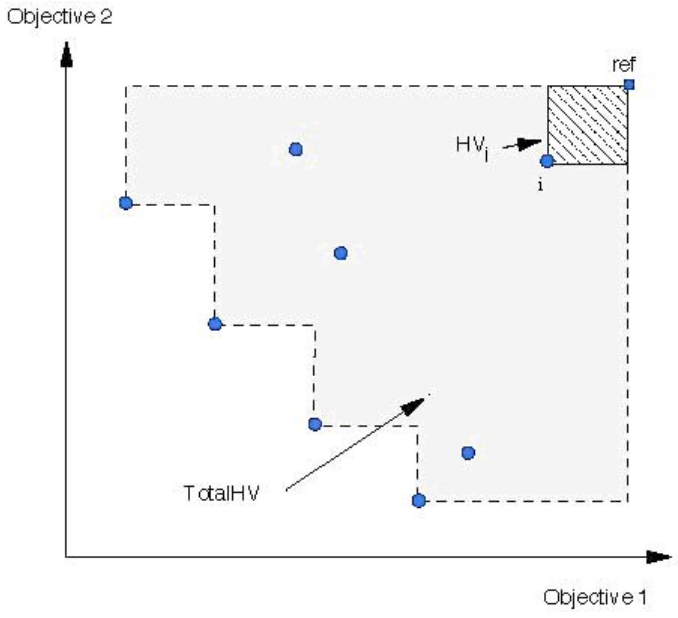


Figure 3: Total Hypervolume (totalHV) of the population and the Hypervolume of the particle i (HV_i). "ref" indicates the reference point

An alternative approach might be to use the Euclidian distance of the solution point from the Pareto-front. In Figure 4 point "A" can be considered to be a distance from the Pareto-front defined by the vector shown. If, for example, a weighted centroid guide particle method had

been used in a MOPSO algorithm (Ireland et al., 2006), these distances would also be readily available for determining ranking, those points furthest from the Pareto-front being considered the "worst".

Finally, it may be possible to rank solutions by comparing them using some aggregated objective function (AOF) that provides a single measure of fitness. The use of simple, weighted-sum AOFs has been shown to have significant difficulties finding all Pareto-dominant solutions, particularly for non-convex Pareto-fronts (Koski, 1985) and their use has been widely deprecated. However, since the solutions of particular interest are not the "best", approximating the Pareto-front, but the "worst" these considerations should not have significant effect. Care should be taken to use an AOF of the form:

$$P_s(\alpha_1,\ldots,\alpha_n;\omega_1,\ldots,\omega_n) = \left(\frac{\omega_1\alpha_1^s + \ldots + \omega_n\alpha_n^s}{\omega_1 + \ldots + \omega_n} \right)^{\frac{1}{s}} \quad (1)$$

where α_i are the solution variables, ω_i are the preference weights, and s is a compensation factor allowing specification of the degree to which compromise solutions are preferred over those in which one objective excels to the exclusion of others (Scott & Antonsson, 2005), protecting such solutions from premature removal.

Which of these various methods to use for specific problems remains an open question.

There also remains the issue of what constitutes a "nearest neighbour" to the points selected for removal in step 4 of the EPSOC algorithm. Current practice has dictated that these neighbouring points be selected on the basis of Euclidian distance in parameter space, with the caveat that members of a large "archive" of preserved solutions – not the archive storing the approximation to the Pareto-front but a large set, often half the population, chosen by rank to make the algorithm greedy – not be removed. For application to multiobjective problems, this latter condition can be altered to protect the archive members only. Use of distance in objective space may also be worthy of consideration.

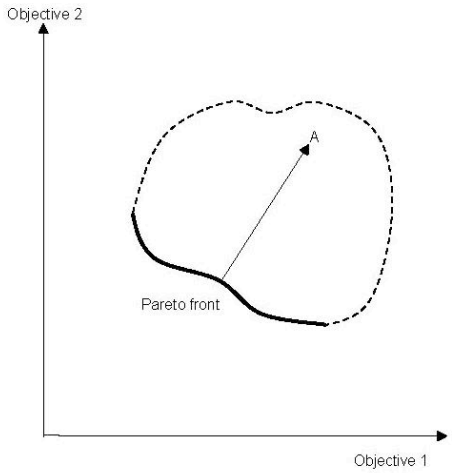


Figure 4: Distance from Pareto-front to determine ranking

COMPUTATIONAL EXAMPLE

EPD has been successfully applied to EO solving combinatorial problems in single objectives (Randall & Lewis, 2006). In this experiment, multi-dimensional knapsack problem (MKP) an extension of the classical knapsack problem, was used as a test case. In it, a mix of items must be chosen that satisfy a series of weighting constraints whilst maximising the collective utility of the items. Equations 2-4 show the 0-1 ILP model.

$$\text{Maximise } \sum_{i=1}^{N} P_i x_i \tag{2}$$

s.t.

$$\sum_{j=1}^{N} w_{ij} x_j \le b_i \qquad \forall i \qquad 1 \le i \le M \tag{3}$$

$$x_i \in \{0,1\} \qquad \forall i \qquad 1 \le i \le N \tag{4}$$

where: x_i is 1 if item i is included in the knapsack, 0 otherwise, P_i is the profit of including item i in the knapsack, N is the total number of items, w_{ij} is the weight of item j in constraint i, M is the number of constraints, and b_i is the total allowable weight according to constraint i.

A number of standard MKP problem instances from the World OR-Library (Beasley, 2007) were used as test cases. The problem instance descriptions are given in Table 1.

Problem Name	*N*	*M*	Optimal Value
mknap1	6	10	3800
mknap2	20	10	6120
mknap3	28	10	12400
mknap4	39	5	10618
mknap5	50	5	6339
mknap6	60	5	6954

Table 1: MKP problem instance descriptions

These problem instances have been used in previous tests of Ant Colony Optimisation (ACO) algorithms (Randall, 2005). In those tests, the ACO algorithm was able to find the global optimum in 4 of the 6 cases, only achieving results of 12380 on mknap3 (0.2%) and 10137 on mknap4 (-4.5%). EO was used by itself, and allowed to run for 500,000 iterations, to ensure achieving the best solution of which the algorithm was capable. Best and median results obtained over a number of trials, and the average actual number of iterations required to obtain the results, are shown in Table 2.

Problem Name	Best result	Iterations taken	Median result	Iterations taken
mknap1	3800	9140	3800	9140
mknap2	6040	166304	5980	7080
mknap3	12180	161089	12180	161089
mknap4	9083	287120	9022	197651
mknap5	6097	172585	6003	31215
mknap6	6650	119523	6583	169988

Table 2: Test results for EO only (© 2006 IEEE)

Using EO with EPD, loop counts for the nested loop were determined empirically, from preliminary experiments, and then set the same for all tests. There is, perhaps, potential for tuning these parameters using self-adaptation (Meyer-Nieberg & Beyer, 2007). However, observation of the convergence history of the hybrid algorithm tends to suggest that the gains from this may not be great. A typical convergence profile (for mknap6) for 100 iterations of EPSOC using 1000, 5000 and 50,000 iterations of EO is shown in Figure 5. The overall shape of each trace is determined by the EPSOC component and shows generally rapid initial convergence, followed by stagnation. Tests of EEO using 1000 iterations of the EPSOC component showed no benefit from allowing it larger numbers of iterations. The effect of changing the number of iterations for the inner, EO, component is primarily apparent in the lower starting values for the traces - by allowing EO to search further, even the first iteration of the hybrid algorithm is improved.

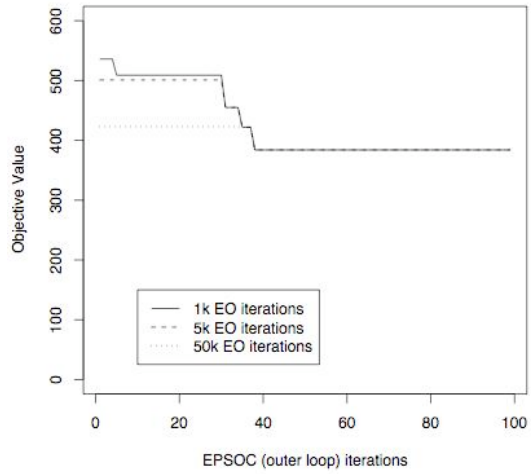


Figure 5: Convergence histories of EO with EPD for mknap6 (© 2006 IEEE)

From Figure 5 it may be observed that the final result is obtained in less than 100 outer loop iterations for any number of inner loop iterations tested, i.e., using at most 100 iterations for the outer and at least 1000 for the inner loop would be adequate to achieve good results in minimal time. For this reason these limits were chosen for use for each of the test cases. The EPSOC component was given a population of 10 to manipulate, i.e., 10 simultaneous EO searches were executed on parallel computing resources at each iteration of the outer loop of the algorithm. The results achieved by the EO algorithm with EPD are given in Table 3.

Problem Name	Value obtained
mknap1	3800
mknap2	6040
mknap3	12210
mknap4	10052
mknap5	6107
mknap6	6570

Table 3: Test results of EO with EPD (© 2006 IEEE)

Comparison of results showed that EO with EPD achieved equal or better results than EO alone on almost all tests cases (in one case it was slightly worse, with a difference of only 1%). On all test cases EO with EPD also achieved its results in less time. On average, EO required 80% more time to obtain generally poorer results. The results using EO with EPD were also compared tests using unmodified ACO. EO with EPD was been found to deliver near-optimal results faster than the existing ACO algorithm, and the relative speed of the algorithm could be expected to improve for larger problem sizes. Further results of these experiments are given elsewhere (Randall & Lewis, 2006).

The experiment described, using EPD with EO on the multi-dimensional knapsack problem, is analogous to applying EPD to a single-objective optimisation algorithm to solve a multiobjective problem using an aggregated objective function. As such, it may prove instructive for this approach to the solution of multiobjective optimisation problems.
To illustrate the use of EPD with Pareto-dominance algorithms for the solution of multi-dimensional problems a standard test problem (Zitzler, Deb & Thiele, 1999) was chosen. The objective functions were of the form:

$$f_1(\mathrm{x}) = x_1$$
$$f_2(\mathrm{x}) = g(\mathrm{x}) \cdot h(f_1, g)$$

where the choice of the *h* and *g* functions dictate the shape of the Pareto-front. For the experiment shown here:

$$g(\mathrm{x}) = 1 + \frac{9}{N-1}\sum_{n=2}^{N} x_n \tag{5}$$

$$h(f_1, g) = 1 - \sqrt{\frac{f_1}{g}} \tag{6}$$

which yields a convex Pareto-front.

The dimension of parameter space, *N*, was 30. A swarm size of 100 was used and the Sigma selection method to determine the guide points. At each iteration of the MOPSO algorithm, points removed from the population by EPD were reinitialised by selecting a member from the archive and initialising a point randomly distributed in a region surrounding the location of the chosen archive member in parameter space.

As the population evolves, non-dominated solutions accumulate in the archive. In a sense, the number of unique solutions in the archive, the “best” solutions found to date, is a measure of

the quality of the algorithm's output. This was found, in this case, to be the chief difference between a standard MOPSO algorithm run with identical conditions, and the MOPSO algorithm with EPD. Traces of the archive size for standard MOPSO and MOPSO with EPD are shown in Figure 6.

When individual solutions in the archives of each algorithm were compared, their locations and objective function values were quite similar. As can be seen in Figure 5, in the early stages there is little to distinguish the performance of the two algorithms. However, MOPSO with EPD gradually starts to accumulate more archive members - more "good" solutions. As more and more points are transferred from "poor" areas closer to the evolving approximation to the Pareto-front, an increasing number find better solutions, enter the non-dominated set – the approximation to the Pareto-front – and are captured in the archive. Toward the end of the program's execution, this trend rapidly accelerates, as MOPSO with EPD fills in more and more of the Pareto-front, yielding a more detailed description of its shape.

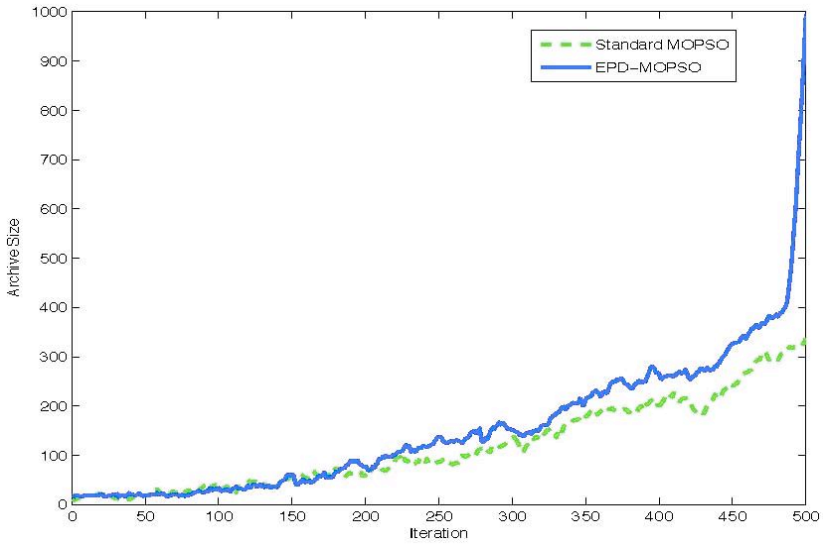


Figure 6: History of archive size for standard MOPSO and MOPSO + EPD

The previous examples show the use of EPD with the EO and MOPSO algorithms. Despite references to the "EPSOC algorithm" it should be noted that EPD is not a new algorithm, competing with other, state-of-the-art algorithms. It is, instead a novel, complementary technique that can be applied to these algorithms. To emphasise this point, the following example demonstrates application of EPD to the widely-used Non-dominated Sorting Genetic Algorithm, NSGA-II (Deb, Agrawal, Pratab & Meyarivan, 2000).

NSGA-II sorts a population of solutions according to their degree of non-domination. That is, all non-dominated solutions in the population are given rank 1. Removing the rank 1 individuals from consideration, those individuals that are non-dominated in the remainder of the population are given rank 2, and so on. Elitism is introduced by comparing the current population at each iteration with the previously-found, best, non-dominated solutions. Conventional processes of tournament selection, recombination and mutation are used to evolve the population.

Since NSGA-II is a population-based algorithm, EPD can be applied directly to the subject population, as in the previous, MOPSO example. At each iteration, *nbad* of the lowest ranked population members are removed, along with their nearest neighbour in solution space. Nearest neighbours are avoided if they fall in the upper half of the ranked population, preserving the elitism of the parent algorithm. The removed individuals are replaced by new individuals initialised close to randomly chosen rank 1 individuals.

This modified algorithm was compared with the un-modified code using five standard, multiobjective test problems (Deb et al., 2000). Details of the problem formulations are given in Table 4. All test case parameters were real-valued. Each problem was run for 10 different, random seeds for both modified and original code. The number of iterations was limited to 50.

Often plots showing several runs are confusing and misleading. Sometimes it is impossible to pick out points from specific run in those plots. In order to illustrate the results of 10 runs, the best (first) and the median (fifth) attainment surfaces (Knowles, 2005) have been plotted. The results of multiple runs build up a set of points that can approximately represent the union of all goals achieved (independently) in the multiple runs. The best attainment surface is a collection of points which build a non-dominated set. The second attainment surface is a set of points which are weakly dominated by the first one. The median surface illustrates an average summary of the obtained results in 10 runs. These attainment surfaces have been produced by diagonal sampling as in Knowles (2005). The attainment surfaces for the best results and median results obtained are plotted in Figures 7a-e and 8a-e, respectively.

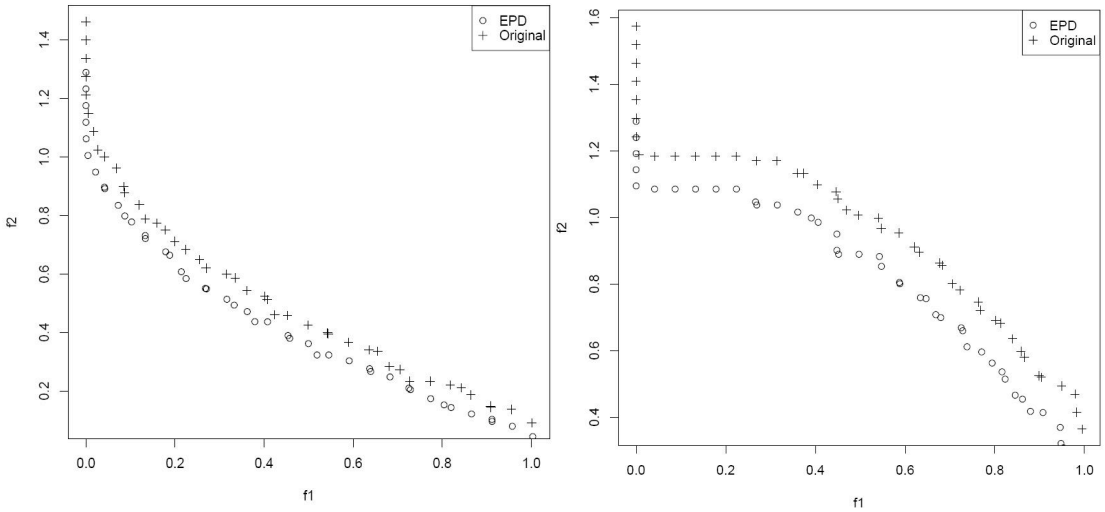
Name	*n*	Variable Bounds	Objective Functions	Type
ZDT1	30	[0,1]	$f_1(\mathbf{x}) = x_1$	Convex
			$f_2(\mathbf{x}) = g(\mathbf{x})[1-\sqrt{x_1/g(\mathbf{x})}]$	
			$g(\mathbf{x}) = 1+9(\sum_{i=2}^{n} x_i)/(n-1)$	
ZDT2	30	[0,1]	$f_1(\mathbf{x}) = x_1$	non-convex
			$f_2(\mathbf{x}) = g(\mathbf{x})[1-(x_1/g(\mathbf{x}))^2]$	
			$g(\mathbf{x}) = 1+9(\sum_{i=2}^{n} x_i)/(n-1)$	
ZDT3	30	[0,1]	$f_1(\mathbf{x}) = x_1$	
			$f_2(\mathbf{x}) = g(\mathbf{x})[1-\sqrt{x_i/g(\mathbf{x})} - \frac{x_1}{g(\mathbf{x})}\sin(10\pi x_1)]$	
			$g(\mathbf{x}) = 1+9(\sum_{i=2}^{n} x_i)/(n-1)$	
ZDT4	10	$x_1 \in [0,1]$	$f_1(\mathbf{x}) = x_1$	non-convex
		$x_i \in [-5,5]$	$f_2(\mathbf{x}) = g(\mathbf{x})[1-\sqrt{x_i/g(\mathbf{x})}]$	

Name	n	Variable Bounds	Objective Functions	Type
		$i = 2,...,n$	$g(\mathbf{x}) = 1 + 10(n-1) + \sum_{i=2}^{n}[x_i^2 - 10\cos(4\pi x_i)]$	
ZDT6	10	[0,1]	$f_1(\mathbf{x}) = 1 - \exp(-4x_1)\sin^6(4\pi x_1)$	non-convex
			$f_2(\mathbf{x}) = g(\mathbf{x})[1 - (f_1(\mathbf{x})/g(\mathbf{x}))^2]$	non-uniform
			$g(\mathbf{x}) = 1 + 9[(\sum_{i=2}^{n} x_i)/(n-1)]^{0.25}$	

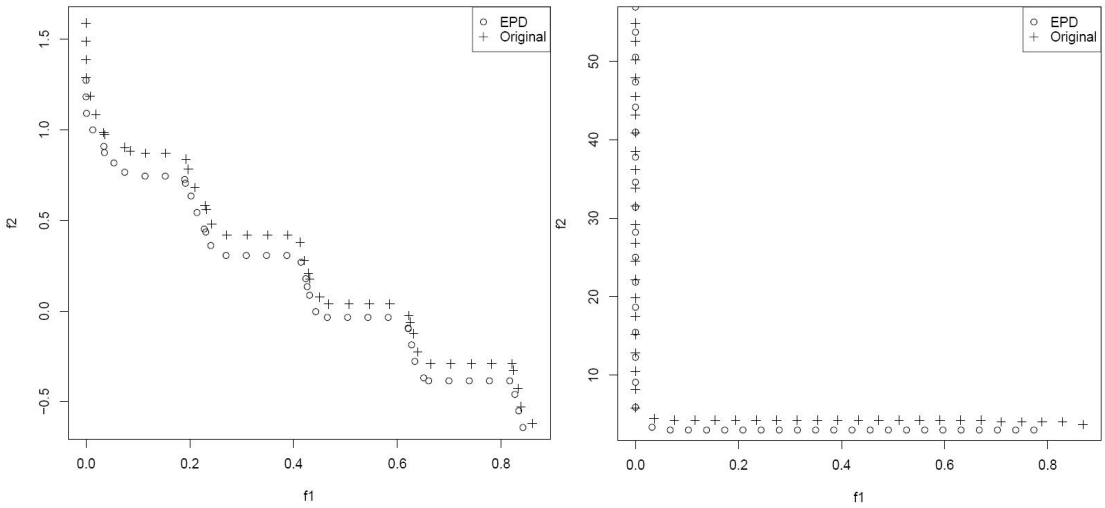
Table 4: The test problems used. All objective functions are to be minimised

From the figures, it may be seen that the algorithm using EPD uniformly delivers better results than the un-modified algorithm. This is, however, for a limited number of iterations. If the two algorithms were allowed to run further, they were found to converge to essentially the same set of Pareto-optimal solutions. EPD appears to confer an advantage in that the set of solutions approximating the Pareto front more rapidly converge toward "reasonably good" solutions. Thereafter, the unmodified algorithm catches up, as the algorithm with EPD does not refine the obtained solutions any more effectively.

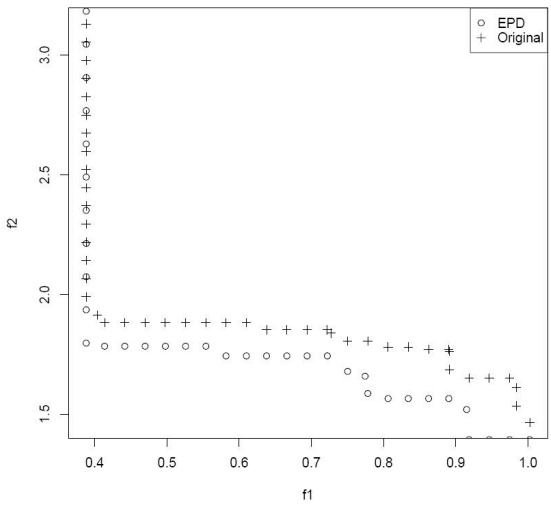
These results are similar to those found in previous applications of the EPSOC algorithm, for example, to a bin-packing problem (Mathieu, 2005). They suggest that hybridisation of algorithms using EPD with local search techniques may be quite profitable, though this remains a topic for future investigation.



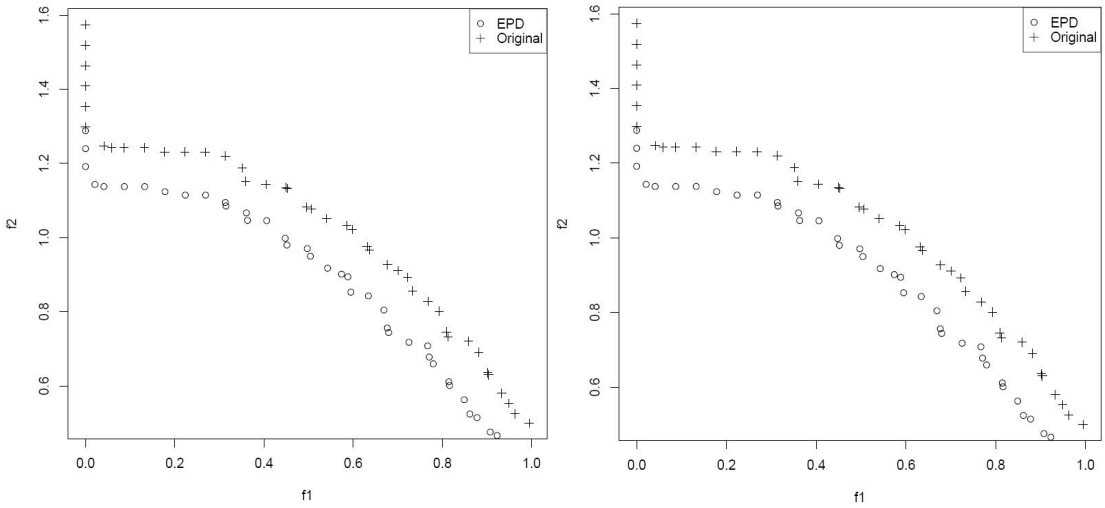
Figures 7a and 7b: Best results for ZDT1 and ZDT2 respectively



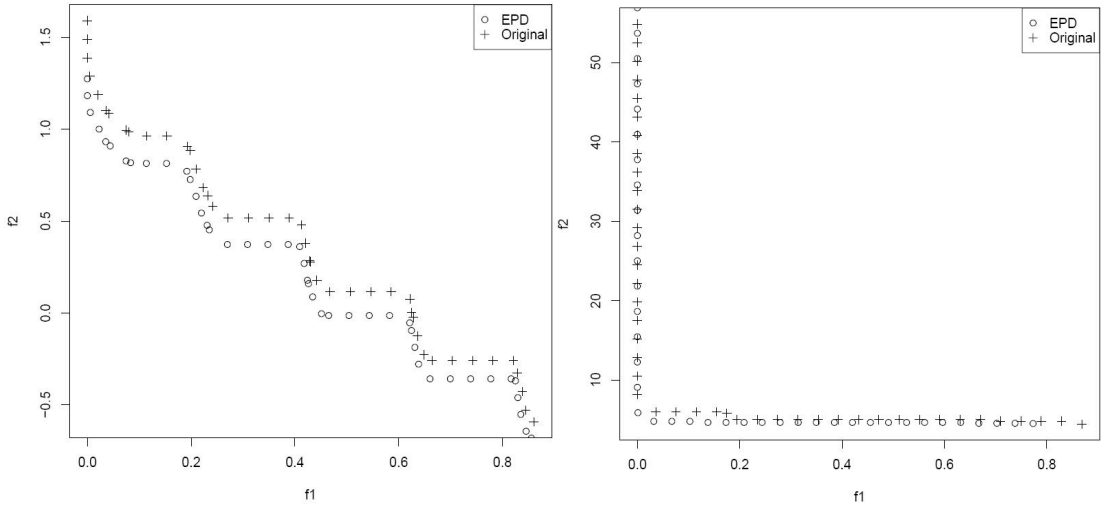
Figures 7c and 7d: Best results for ZDT3 and ZDT4 respectively



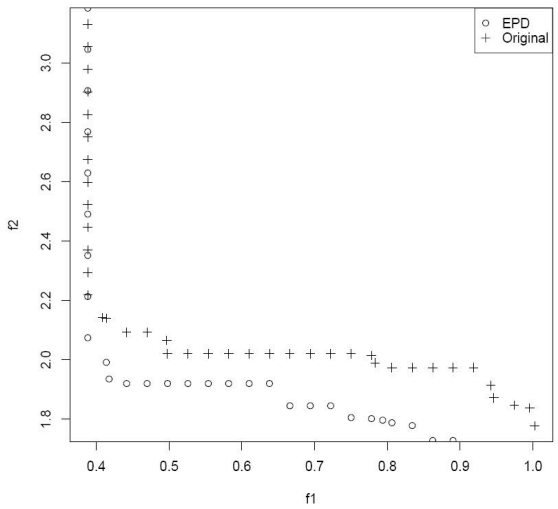
Figures 7e: Best results for ZDT6



Figures 8a and 8b: Median results for ZDT1 and ZDT2 respectively



Figures 8c and 8d: Median results for ZDT3 and ZDT4 respectively



Figures 8e: Median results for ZDT6

CONCLUDING REMARKS AND FUTURE RESEARCH DIRECTIONS

The use of evolutionary population dynamics (EPD) as a controlling metaheuristic for population-based optimisation algorithms is a recent development. As briefly outlined in this chapter, there are a number of open questions regarding the most effective means for its application. This work has sought to describe some of the fundamental concepts underpinning this novel paradigm, the rationale behind its use, and some practical suggestions for its implementation. Some of the potential benefits of its use – better solutions derived more rapidly – have been demonstrated from preliminary experimental results. As has been shown, this concept can be applied to both continuous and combinatorial problems for single-valued and multiobjective problems. While designed for use with population-based algorithms, a suggested methodology for applying the concept to other types of algorithms has also been described, and its efficacy demonstrated.

This is a newly emerging approach and, while early results have been very encouraging, considerable work remains to examine and understand the performance, emergent behaviour and expected outcomes of its use in an expanding range of applications. EPD therefore has a great scope for further improvement and refinement. We believe that the following lines of investigation could prove profitable:

- *Methods of selection of "poor" solutions for removal* – As outlined in the body of the chapter, three alternatives can be envisaged:
  1. Comparison using hypervolume metrics
  2. Measurement of Euclidean distance to the Pareto-front (or some "Utopia Point", as utilised in several performance metrics)
  3. Ranking of solutions by use of an Aggregated Objective Function (AOF)

  In addition to these three, the ranking methods of the algorithms themselves can be used. For example, NSGA-II sorts individuals in its population by means of degree of domination. This ranking was implicitly used in the computational experiments applying EPD to NSGA-II described.
- *Methods for determination of "nearest neighbours"* – during selection of solutions for removal - At present a large (half the entire population) subset of solutions is protected from removal as part of the greedy elitism. The effect of reducing this to members of the archive approximating the Pareto optimal set, across a range of algorithms and test problems, has yet to be determined.
- *Methods of re-initialisation of points removed from the population at each iteration by the application of EPD* – The experiments described here use simple methods of random re-initialisation of trial solutions in regions "close" to known, good solutions. These approaches were chosen on the basis of limited, empirical comparison with uniformly random re-initialisation. More sophisticated approaches, and a more systematic comparison of emergent behaviour, may prove beneficial.
- *The possible benefits of the use of self-adaptation* – Self adaptation is when an algorithm optimises its own control parameters while it is solving a problem. In the case of EPD, the possible parameters for include the number of iterations of inner and outer loops; number of *nbad* trial solutions to be removed (where appropriate); and the number of neighbouring points.

- *The hybridisation of EPD-manipulated algorithms with techniques of local search* – This promises significant advantages in a number of areas, particularly in achieving computational efficiency.

ACKNOWLEDGEMENTS

The authors wish to thank David Ireland for his assistance in generating the experimental results described in this work.

REFERENCES

Alvarez-Benitez, J., Everson, R., & Fieldsend, J. (2005). A MOPSO algorithm based exclusively on Pareto dominance concepts. In C. Coello-Coello, A. Aguirre & E. Zitzler (Eds.), *Evolutionary Multi-Criterion Optimization: Vol. 3410*. *Lecture Notes in Computer Science* (pp. 459–473). Berlin: Springer-Verlag.

Angus, D. (2006). Crowding population-based ant colony optimisation for the multi-objective travelling salesman problem. In P. Sloot, G. van Albada, M. Bubak, & A. Referthen (Eds.), *2nd IEEE International e-Science and Grid Computing Conference (Workshop on Biologically-inspired Optimisation Methods for Parallel and Distributed Architectures: Algorithms, Systems and Applications)*. Piscataway: IEEE-Press.

Bak, P., Tang, C. & Wiesenfeld, K. (1987). Self-organized criticality: an explanation of 1/f noise, *Physical Review Letters*, *59*, 381–384.

Bak, P. & Sneppen, K. (1993). Punctuated equilibrium and criticality in a simple model of evolution, *Physical Review Letters*, *71*, 4083–4086.

Bak, P. (1996). *How nature works*. New York: Springer-Verlag.

Beasley, J. (2007). *OR-library*, Retrieved January 1, 2007, from http://people.brunel.ac.uk/mastjjb/jeb/info.html.

Boettcher, S., & Percus, A. (1999). Extremal optimization: Methods derived from co-evolution, In W. Banzhaf, J. Daida, A. Eiben, M. Garzon, V. Honavar, M. Jakiela, & R. Smith (Eds.), *Genetic and Evolutional Computation Conference* (pp. 825-832). San Francisco: Morgan Kaufmann.

Boettcher, S., & Percus, A. (2000). Nature's way of optimizing, *Artificial Intelligence*, *119*, 275–286.

Boettcher, S., & Percus, A. (2003). Extremal optimization: An evolutionary local search algorithm", In H. Bhargava & N. Ye (Eds.), *8th INFORMS Computer Society Conference: Vol. 21. Interfaces in Computer Science and Operations Research* (pp. 61-78). Norwell, MA: Kluwer Academic Publishers.

Branke, J., & Mostaghim, S. (2006). About selecting the personal best in multi-objective particle swarm optimization. In T. Runarsson, H. Beyer, E. Burke, J. Merelo Guervos, L. Darrell Whitley, & X. Yao, (Eds.), *Parallel Problem Solving from Nature: Vol. 4193. Lecture Notes in Computer Science* (pp. 523–532). Berlin: Springer-Verlag.

Coello Coello, C., & Lechuga, M. (2002). MOPSO: A proposal for multiple objective particle swarm optimization. In D. Fogel, M. El-Sharkawi, X. Yao, G. Greenwood, H. Iba. P. Marrow, & M. Shackleton (Eds.), *Congress on Evolutionary Computation* (pp. 1051–1056). Piscataway: IEEE-Press.

Coello Coello, C., Van Veldhuizen, D., & Lamont, G. (2002). *Evolutionary algorithms for solving multi-objective problems*. Norwell, MA: Kluwer Academic Publishers.

Deb, K., Agrawal, S., Pratab, A., & Meyarivan, T. (2000). *A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II*. (KanGAL Rep. No. 200001). Kanpur, India: Indian Institute of Technology.

Deb, K. (2001). *Multi-objective optimization using evolutionary algorithms*. New York: John Wiley and Sons.

Doerner, K., Gutjahr, W., Hartl, R., & Strauss, C. (2004). Pareto ant colony optimization: A metaheuristic approach to multiobjective portfolio selection. *Annals of Operations Research*, *131*, 79–99.

Dorigo, M., & Di Caro, G. (1999). The ant colony optimization meta-heuristic. In D. Corne, M. Dorigo, & F. Glover (Eds.), *New Ideas in Optimization* (pp. 11–32). London: McGraw-Hill.

Eberhart, R. & Kennedy, J. (1995). A new optimizer using particles swarm theory. In *6th International Symposium on Micro Machine and Human Science* (pp. 39–43). Piscataway: IEEE-Press.

Engelbrecht, A. (2005). *Fundamentals of computational swarm intelligence*. New York: John Wiley and Sons.

Fieldsend, J., & Singh, S. (2002). A multi-objective algorithm based upon particle swarm optimisation, an efficient data structure and turbulence. In X. Tao (Ed.), *The U.K. Workshop on Computational Intelligence* (pp. 34–44).

Fogel, L. (1962). Autonomous automata, *Industrial Research*, *4*, 14–19.

Galski, R., de Sousa, F., Ramos, F., & Muraoka, I. (2007). Spacecraft thermal design with the generalized extremal optimization algorithm. *Inverse Problems in Science and Engineering*, *15*, 61–75.

Gambardella, L., Taillard, E., & Agazzi, G. (1999). MACS-VRPTW - A multiple ant colony system for vehicle routing problems with time windows. In D. Corne, M. Dorigo, & F. Glover (Eds.), *New Ideas in Optimization* (pp. 63–76). London: McGraw-Hill.

Garcia-Martinez, C., Cordón, O., & Herrera, F. (2004). An empirical analysis of multiple objective ant colony optimization algorithms for the bi-criteria TSP. In M. Dorigo, M. Birattari, C. Blum, L. Gambardella, F. Mondada, & T. Stützle (Eds.) *4th International Workshop on Ant Colony Optimization and Swarm Intelligence: Vol. 3172. Lecture Notes in Computer Science* (pp. 61–72), Berlin: Springer-Verlag.

Garcia-Martinez, C., Cordón, O., & Herrera, F. (2007). A taxonomy and an empirical analysis of multiple objective ant colony optimization algorithms for bi-criteria TSP. *European Journal of Operational Research*, *180*, 116–148.

Goldberg, D. (1989). *Genetic algorithms in search, optimization and machine learning*. Reading, MA: Addison Wesley.

Guntsch, M., & Middendorf, M. (2003). Solving multi-criteria optimization problems with population-based ACO, In G. Goos, J. Hartmanis, & J. van Leeuwen (Eds.), *Second International Conference on Evolutionary Multi-Criterion Optimization: Vol. 2632. Lecture Notes in Computer Science* (pp. 464–478). Berlin: Springer-Verlag.

Hu, X., & Eberhart, R. (2002). Multiobjective optimization using dynamic neighborhood particle swarm optimization. In D. Fogel, M. El-Sharkawi, X. Yao, G. Greenwood, H. Iba. P. Marrow, & M. Shackleton (Eds.), *Congress on Evolutionary Computation* (pp. 1677–1681). Piscataway: IEEE-Press.

Iredi, S., Merkle, D., & Middendorf, M. (2001). Bi-criterion optimization with multi colony ant algorithms. In E. Zitzler, K. Deb, L. Thiele, C. Coello Coello, & D. Corne (Eds.), *First International Conference on Evolutionary Multi-Criterion Optimization: Vol. 1993. Lecture Notes in Computer Science* (pp. 359–372). Berlin: Springer-Verlag.

Ireland, D., Lewis, A., Mostaghim, S., & Lu, J. (2006). Hybrid particle guide selection methods in multi-objective particle swarm optimization. In P. Sloot, G. van Albada, M. Bubak, & A. Referthen (Eds.), *2nd International e-Science and Grid Computing Conference (Workshop on Biologically-inspired Optimisation Methods for Parallel and Distributed Architectures: Algorithms, Systems and Applications)*. Piscataway: IEEE-Press.

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. In *International Conference on Neural Networks* (pp. 1942–1948). Piscataway: IEEE-Press.

Knowles, J. (2005). A summary-attainment-surface plotting method for visualizing the performance of stochastic multiobjective optimizers. In *5th International Conference on Intelligent Systems Design and Applications* (pp. 552-557).

Koski, J. (1985). Defectiveness of weighting method in multicriterion optimization of structures. *Communications in Applied Numerical Methods*, *1*, 333–337.

Lewis, A., Abramson, D., & Peachey, T. (2003). An evolutionary programming algorithm for automatic engineering design. In R. Wyrzykowski, J. Dongarra, M. Paprzycki, & J. Wasniewski (Eds.), *5th International Conference on Parallel Processing and Applied Mathematics: Vol. 3019. Lecture Notes in Computer Science* (pp. 586-594). Berlin: Springer-Verlag.

Mariano, C., & Morales, E. (1999). *A multiple objective Ant-Q algorithm for the design of water distribution irrigation networks*, (Tech. Rep. HC-9904). Mexico City: Instituto Mexicano de Tecnologia del Agua.

Mathieu, O. (2005). *Utilisation d'algorithmes évolutionnaires dans le cadre des problèmes d'empaquetage d'objets*, Unpublished master's thesis, Facultes Universitaires Notre-Dame de la Paix Namur, Brussels.

Meyer-Nieberg, S., & Beyer, H. (2007). Self-adaptation in evolutionary algorithms. In F. Lobo, C. Lima, & Z. Michalewicz (Eds.), *Parameter Settings in Evolutionary Algorithms: Vol. 54. Studies in Computational Intelligence* (pp. 47-76). Berlin: Springer-Verlag.

Mostaghim, S. Teich, J. (2003). Strategies for finding good local guides in multi-objective particle swarm optimization. In *Swarm Intelligence Symposium* (pp. 26–33). Piscataway: IEEE-Press.

Mostaghim, S. (2005). *Multi-objective Evolutionary Algorithms: Data structures, Convergence and, Diversity*. Doctoral thesis, University of Paderborn, Paderborn.

Mostaghim, S., Branke, J., & Schmeck, H. (2006). *Multi-objective particle swarm optimization on computer grids* (Tech. Rep. No. 502). Karlsruhe, Germany: University of Karlsruhem, AIFB Institute.

Mostaghim, S., & Halter, W. (2006). Bilevel optimization of multi-component silicate melts using particle swarm optimization. In D. Fogel (Ed.), *Congress on Evolutionary Computation* (pp. 4383-4390). Piscataway: IEEE-Press.

Parsopoulos, K., & Vrahatis, M. (2002). Particle swarm optimization method in multiobjective problems, In B. Panda (Ed.), *Symposium on Applied Computing* (pp. 603–607). New York: ACM Press.

Purshouse, R., & Fleming, P. (2003). Evolutionary multi-objective optimisation: An exploratory analysis. In R. Sarker, R. Reynolds, H. Abbass, K. Tan, B. McKay, D. Essam, & T, Gedeon (Eds.), *Congress on Evolutionary Computation* (pp. 2066–2073). Piscataway: IEEE-Press.

Randall, M. (2005). A dynamic optimisation approach for ant colony optimisation using the multidimensional knapsack problem, In H. Abbass, T. Bossamaier, & J. Wiles (Eds.), *Recent Advances in Artificial Life* (pp. 215 – 226). Singapore: World Scientific.

Randall, M., & Lewis, A. (2006). An extended extremal optimisation model for parallel architectures", In P. Sloot, G. van Albada, M. Bubak, & A. Referthen (Eds.), *2nd International e-Science and Grid Computing Conference (Workshop on Biologically-inspired Optimisation Methods for Parallel and Distributed Architectures: Algorithms, Systems and Applications)*. Piscataway: IEEE-Press.

Reyes-Sierra, M., & Coello Coello, C. (2006). Multi-objective particle swarm optimizers: A survey of the state-of-the-art. *International Journal of Computational Intelligence Research, 2*, 287-308.

Scott, M., & Antonsson, E. (2005). Compensation and weights for trade-offs in engineering design: Beyond the weighted sum. *Journal of Mechanical Design*, *127*, 1045–1055.

Zitzler, E. (1999). *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*. Doctoral thesis, Swiss Federal Institute of Technology (ETH).

Zitzler, E., Deb, K., & Thiele, L. (1999). Comparison of multiobjective evolutionary algorithms on test functions of different difficulty. In A. Wu (Ed.), *Genetic and Evolutionary Computation Conference (Workshop Program)* (pp. 121–122).

ADDITIONAL READING

This chapter draws on a number of different fields. In order to gain a greater understanding of these, the following references are provided over and beyond those in the References list; metaheuristic search algorithms overview (Back, Fogel & Michalewicz, 1999; Bonabeau, Dorigo & Theraulaz, 1999; Ebehart, Simpson & Dobbins, 1996; Eberhart, Shi & Kennedy, 2001; Michalewicz & Fogel, 2004), ACO (Dorigo, Maniezzo & Colorni, 1996; Dorigo & Stützle, 2004; Dorigo & Stützle, 2003; Guntsch & Middendorf, 2002; López-Ibáñez, Paquete & Stützle, 2004), EO (Boettcher, 2000; Boettcher & Percus, 2001), PSO (Clerc, 2006; Kennedy & Eberhart, 1999), multiobjective optimisation (Coello Coello, 2006a; Coello Coello, 1999; Coello Coello, 2000; Coello Coello, 2006b; Collette & Siarry, 2004; Deb, 2000; Deb & Jain, 2004; Deb, Thiele, Laumanns & Zitzler, 2005; Zitzler & Thiele, 1999).

Back, T., Fogel, D., & Michalewicz, Z. (Eds.). (1997). *Handbook of Evolutionary Computation*. Oxford: Oxford University Press.

Bonabeau, E., Dorigo, M., & Theraulaz, G. (1999). *Swarm intelligence*. Oxford: Oxford University Press.

Boettcher, S. (2000). Extremal Optimization - Heuristics via co-evolutionary avalanches. *Computing in Science & Engineering*, *2*, 75-82.

Boettcher, S., & Percus, A, (2001). Optimization with extremal dynamics. *Physics Review Letters*, *86*, 5211–5214.

Clerc, M. (2006). *Particle Swarm Optimization*. UK: ISTE Publishing Company.

Coello Coello, C. (1999). A comprehensive survey of evolutionary-based multiobjective optimization techniques. *Knowledge and Information Systems*, *1*, 269-308.

Coello Coello, C. (2000). Handling preferences in evolutionary multiobjective optimization: A survey. In *Congress on Evolutionary Computation* (pp. 30-37). Piscataway: IEEE Computer Society Press.

Coello Coello, C., Toscano Pulido, G. & Salazar Lechuga, S. (2002). An extension of particle swarm optimization that can handle multiple objectives. In *The Workshop on Multiple Objective Metaheuristics*.

Coello Coello, C. (2006a). 20 years of evolutionary multi-objective optimization: What has been done and what remains to be done. In G. Yen & D. Fogel (Eds.), *Computational Intelligence: Principles and Practice* (pp. 73-88). Piscataway: IEEE-Press.

Coello Coello, C. (2006b). The EMOO repository: A resource for doing research in evolutionary multiobjective optimization, *IEEE Computational Intelligence Magazine*, *1*, 37-45.

Collette, Y., & Siarry, P. (2004). *Multiobjective Optimization: Principles and Case Studies (Decision Engineering)*. Berlin: Springer-Verlag.

Deb, K. (2000). Multi-objective evolutionary optimization: Past, present and future. In I. Parmee (Ed.), *Evolutionary Design and Manufacture* (pp. 225-236). Berlin: Springer-Verlag.

Deb, K., & Jain, S. (2004). Evaluating evolutionary multi-objective optimization algorithms using running performance metrics. In K. Tan, M. Lim, X. Yao, & L. Wang (Eds.), *Recent Advances in Simulated Evolution and Learning* (pp. 307-326). Singapore: World Scientific Publishers.

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

Dorigo, M., Maniezzo, V., & Colorni, A. (1996). The ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man and Cybernetics – Part B*, *26*, 29-41.

Dorigo, M., & Stützle, T. (2003). The ant colony Optimization metaheuristic: Algorithms, applications, and advances. In F. Glover & G. Kochenberger (Eds.), *Handbook of metaheuristics* (pp. 251-285). Norwell, MA: Kluwer Academic Publishers.

Dorigo, M., & Stützle, T. (2004). *Ant Colony Optimization*. Cambridge, MA: MIT Press.

Englebrecht, A. (2006). *Fundamentals of Computational Swarm Intelligence*. New Jersey: Wiley.

Eberhart, R., Simpson, P., & Dobbins, R. (1996). *Computational Intelligence PC Tools*. 1st edition, Boston, MA: Academic Press Professional, 1996.

Eberhart, R., Shi, Y., & Kennedy, J. (2001). *Swarm Intelligence*. San Fransisco: Morgan Kauffman.

Guntsch, M., & Middendorf, M. (2002). A population based approach for ACO. In S. Cagnoni, J. Gottlieb, E. Hart, M. Middendorf, & G. Raidl (Eds.), *Applications of Evolutionary Computing: Vol. 2279. Lecture Notes in Computer Science* (pp. 72–81). Berlin: Springer-Verlag.

Kennedy, J., & Eberhart, R. (1999). The particle swarm: Social adaptation in information-processing systems. In D. Corne, M. Dorigo, & F. Glover (Eds.) *New ideas in optimization* (pp. 379–387). London: McGraw-Hill.

López-Ibáñez, M., Paquete, L., & Stützle, T. (2004). On the design of ACO for the biobjective quadratic assignment problem. In M. Dorigo, M. Birattari, C. Blum, L. Gambardella, F. Mondada, & T. Stützle (Eds.), *4th International Workshop on Ant Colony Optimization and Swarm Intelligence: Vol. 3172. Lecture Notes in Computer Science* (pp. 61–72), Berlin: Springer-Verlag.

Michalewicz, Z., & Fogel, D. (2004). *How to solve it: Modern heuristics*. Berlin: Springer-Verlag.

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

[i] PACO uses a population of solutions to derive its pheromone matrix, rather than directly from the solution components.

[ii] In terms of ACO, PACO and its variants are the most suitable for EPD. This is because we can apply its mechanisms to the population of pheromone deriving solutions.