Multi-Objective Probability Collectives

Antony Waldock[1] and David Corne[2]

[1] Advanced Technology Centre, BAE Systems, Bristol, UK
antony.waldock@baesystems.com
[2] School of MACS, Heriot-Watt University, Edinburgh, UK
dwcorne@macs.hw.ac.uk

Abstract. We describe and evaluate a multi-objective optimisation (MOO) algorithm that works within the Probability Collectives (PC) optimisation framework. PC is an alternative approach to optimization where the optimization process focusses on finding an ideal distribution over the solution space rather than an ideal solution. We describe one way in which MOO can be done in the PC framework, via using a Pareto-based ranking strategy as a single objective. We partially evaluate this via testing on a number of problems, and compare the results with state of the art alternatives. We find that this first multi-objective probability collectives (MOPC) approach performs competitively, indicating both clear promise, and clear room for improvement.

1 Introduction

1.1 Multi-Objective Optimisation

Multi-objective optimisation (MOO) continues to gain increasing attention [7], as it becomes recognised that (a) a large number of real-world optimisation problems are multi-objective; (b) the classical simplifying approach of combining many objectives into one has several drawbacks [4]; (c) several efficient and effective methods now exist that address MOO in a more principled way (e.g. [32, 8, 21]).

A MOO problem is formally posed as $\arg\min_{\boldsymbol{x}\in X} G_m(\boldsymbol{x})$, where $G_m(\boldsymbol{x})$ is an objective function and $\boldsymbol{x}$ is defined as a vector of decision variables (or a solution) in the form $\boldsymbol{x} = (x_1, x_2.., x_N)$ from the set of solutions X. The aim is to find the *Pareto* set which contains all the solutions that are not dominated by any other solution. A solution $\boldsymbol{x_1}$ is said to be dominated by $\boldsymbol{x_2}$, if and only if, $\boldsymbol{x_1}$ is as good as $\boldsymbol{x_2}$ in all objectives and $\boldsymbol{x_1}$ is strictly better then $\boldsymbol{x_2}$ in at least one objective. A distinguishing feature of MOO is that the target is a set of solutions rather than a single 'best'.

The most effective MOO approaches to date are generally regarded to be MOEAs (multi-objective evolutionary algorithms). EAs naturally maintain diversity (by working with a population of solutions), and many additional techniques exist (e.g. [11, 14, 19]). MOEAs encompass a broad family of approaches to MOO; particularly successful among them are those based on particle swarm

optimisation (PSO) [17, 20, 24], and others based on decomposing the MOO problem into several different single-objective problems [16, 10]. In the former approach, PSO is combined with the use of measures to maintain a diverse set of 'targets', usually ensuring these are spread well across the developing Pareto set. In the latter approach, the idea is to exploit the relationship between MOO and single objective problems. Roughly speaking, different regions of the Pareto set are optima of different weighted sums of the original objectives; such methods simultaneously progress several different single-objective searches, together aiming to cover the Pareto set.

Meanwhile, an alternative framework for optimisation is Probability Collectives (PC) [3, 29], distinguished by a focus on finding an ideal distribution over solution space, rather than an ideal solution. This has similarities to estimation of distribution algorithms (EDAs). However, EDAs operate by continually updating a distribution guided by the results of sampling, with the goal of finding optimal samples. PC, on the other hand, optimizes the distribution itself, and this is reflected in a principled approach to the way that sample evaluations influence the distribution. PC-based optimization is naturally well-suited to maintaining diversity, as well as handling uncertainty and noise. Meanwhile, PC shares with EDAs the clear advantages of using a distribution as the focus of search, rather than just concentrating (and perhaps being misled by) the 'survivors' represented by a current set of samples.

PC clearly has much potential for use in MOO. In this paper, we explore an initial approach to adapting the PC framework for MOO, by formulating the problem using a proxy single-objective, in a similar vein to PSO. It is revealing to see how well this preliminary approach fares against state of the art MOO methods. In the remainder, section 2 introduces Probability Collectives, section 3 introduces the proposed algorithm: Multi-Objective Probability Collectives (MOPC), the experimental set-up and results are in section 4, and we conclude in section 5 with conclusions and future work.

2 Probability Collectives

Probability Collectives (PC) is a framework for black-box optimisation with deep theoretical connections to game theory, statistical physics [26], and optimisation [28]. It has been applied so far to problems, ranging from sensor management [25] to distributed flight control [2].

Typically a single-objective problem is solved by manipulating a vector of decision variables $\boldsymbol{x}$ in an attempt to minimise a scalar function ($G(\boldsymbol{x})$). In PC, however, optimisation is performed on probability distributions $q(\boldsymbol{x})$ over the decision variables, seeking a distribution highly peaked around values of the decision variables that minimise $G(\boldsymbol{x})$. This approach has various demonstrated advantages; it is: easily parallelised (each variable's distribution can be updated independently [18]); applicable to continuous, discrete, or arbitrary spaces [3]; robust to noise and irregularity [9]; provides sensitivity information – a variable with a peaky distribution can be deemed more important than one with a broad

distribution. The PC framework formally defines optimisation as in Equation 1.

$$\arg\min_{q_\theta \in \mathscr{P}} \mathbb{E}_{q_\theta}(G(\boldsymbol{x})) \tag{1}$$

where q_θ is a parametric distribution over the decision variables $\boldsymbol{x}$ in the set of all possible distributions $\mathscr{P}$, minimising the expectation $\mathbb{E}_{q_\theta}(G(\boldsymbol{x}))$.

PC does not prescribe a specific approach for minimising this expectation. Many alternatives are discussed in [27]. From hereon, we follow one such approach in which, based on considering the expectation of all possible distributions[22], $\mathbb{E}_{\mathscr{P}}(G(\boldsymbol{x}))$, one solution is the point-wise limit of the Boltzmann distributions shown in Equation 2.

$$p^*(x) = \lim_{\beta \to \infty} p^\beta(\boldsymbol{x}) \tag{2}$$

where $p^\beta(\boldsymbol{x})$ is defined as $\exp[-\beta G(\boldsymbol{x})]$. So, as β tends towards ∞ the distributions of p^β become peaked around the solutions that minimise $G(\boldsymbol{x})$. To find $p^*(\boldsymbol{x})$, a parametrised distribution q_θ is used to approximate the Boltzmann distributions, and fitted to the Boltzmann distribution p^β by minimising the Kullback-Leibler (KL) divergence [13] in Equation 3.

$$\mathbb{E}_{q_\theta}(G(\boldsymbol{x})) = -KL(p^\beta \| q_\theta) \qquad = -\int p^\beta \ln\left(\frac{p^\beta}{q_\theta}\right) dx \tag{3}$$

By minimising the KL Divergence, q_θ will approximate the "target" of $p^\beta(\boldsymbol{x})$. The β term is used as a regularization parameter controlling the evolution of the distribution towards areas of decision space minimising $G(\boldsymbol{x})$. The high-level algorithm we use can now be presented in Alg. 1. We formulate the minimisation

Algorithm 1 PC Optimisation

1: Initialise β to be β_{min}
2: Initialise the number of evaluations to 0
3: **repeat**
4: Draw a set D from X using uniform distribution on the first run or q_θ thereafter
5: Evaluate $G(\boldsymbol{x})$ for each sample drawn
6: Find q_θ by minimising the KL Divergence
7: Update β
8: Update evaluations
9: **until** (evaluations > maximum evaluations)

of KL divergence as cross-entropy minimisation [23] using a single multivariate Gaussian density with mean μ and covariance σ. Means $\boldsymbol{\mu}$ and co-variances $\boldsymbol{\sigma}$ are updated from samples as follows:

$$\boldsymbol{\mu} = \frac{\sum_D s^i \boldsymbol{x}^i}{\sum_D s^i}; \qquad \boldsymbol{\sigma} = \frac{\sum_D s^i (\boldsymbol{x}^i - \boldsymbol{\mu})(\boldsymbol{x}^i - \boldsymbol{\mu})^T}{\sum_D s^i} \tag{4}$$

where s^i is defined as $p(\boldsymbol{x}^i)$ and $\boldsymbol{x}^i$ is the i^{th} sample in the sample set D. Recall that $p(\boldsymbol{x}_i)$ is defined using a Boltzmann distribution $\exp[-\beta G(\boldsymbol{x}^i)]$.

Note that there are close parallels with Expectation Maximisation (EM). The difference from EM is the inclusion of the s^i term which is driven by β included in the Boltzmann distribution. We can regard β as parametrising a trade-off; when small, the parametric distribution tends to fit the samples, regardless of $G(x)$. As β tends towards infinity, the focus shifts towards samples with the best $G(x)$ by producing a highly peaked distribution.

3 Multi-Objective Probability Collectives

Multi-Objective Probability Collectives (MOPC) is implemented here by using a single-objective 'proxy' – i.e. a single-objective score that attempts to evaluate a solution's quality for inclusion in the Pareto set. For this purpose we adopt the maximin function from [17]. There are alternatives, such as Average Ranking or others detailed in [5, 1], however our choice of maximin is based on promising performance within a PSO based MOO strategy [17]. Broad pseudocode for MOPC is shown in Alg. 2. In MOPC, β is replaced by T, defined as $\frac{1}{\beta}$ for convenience. Firstly, MOPC initialises an archive A to keep track of the developing Pareto set, which in our current implementation is maintained in the same way as the Crowding Archive used in NSGAII [8]. Next, MOPC initialises T and a counter for number of evaluations. MOPC calculates the decay rate for T based on T_{start}, T_{end}, the maximum number of evaluations allowed E, the number of particles $|P|$ and the number of samples taken on each iteration $|D|$.

Algorithm 2 MOPC Optimisation

1: Initialise the archive set A to empty, T to T_{start} and calculate T_{decay} and the number of evaluations to 0
2: Initialise the set of MOPC Particles P
3: **repeat**
4: **for all** MOPC Particles **do**
5: Update MOPC Particle using A (see algorithm 3)
6: Increment the number of evaluations taken
7: **end for**
8: **if** $(T > T_{end})$ **then** Decrement T **end if**
9: **until** (evaluations > maximum evaluations)
10: Output the non-dominated set from archive set A

$$T_{decay} = \frac{T_{end}}{T_{start}}^{\frac{|P|*|D|}{E}} \tag{5}$$

MOPC then repeatedly updates each of the particle's parametric distributions while reducing T until the maximum number of evaluations is reached. The

output of MOPC is the archive A. Details of line 7 of Alg. 2 are given by Alg. 3. In Alg. 3, each particle performs its own PC optimisation. First, samples are

Algorithm 3 Update MOPC Particle

1: **if** first run **then**
2: Draw and evaluate a set of samples D from X using a uniform distribution for the first run and q_θ thereafter
3: **end if**
4: Add the samples taken in D to a local cache L
5: Calculate maximin for the members of $L \cup A$
6: Find the new q_θ (using L) by minimising the KL Divergence (Eqs. 3, 4)
7: Add the samples from D that are not dominated to the archive A

drawn from q_θ (initially, a uniform distribution), and then added to local cache L to enable previously used samples to be reused when upating the parametric distribution; members of $L \cup A$ are then evaluated using $f_{maximin}$ (Eqn 6):

$$f_{maximin}(x) = \max_{j=1,2..|L\cup A|;x\neq x^j}(\min_{m=1,..,M}(G_i(x) - G_i(x^j))) \tag{6}$$

where m is the objective function, x^j is the j^{th} sample in the set and $f_{maximin}(x)$ is the fitness value for the sample x. This returns a value indicating how much x is dominated by the other samples. When $f_{maximin}(x)$ is $< 0| = 0| > 0$, x is non-dominated | weakly-dominated | dominated. Hence, the distribution is driven towards non-dominated solutions. Distribution q_θ is the updated, and finally, non-dominated samples from D are archived.

4 Experiments and Results

We first explore the behaviour of MOPC on the DEB2 problem in [6], which was specificially designed to present difficult challenges for MOO, arising from multimodality (multiple suboptimal Pareto sets) and deception (structures in the search space that tend to mislead optimisation algorithms towards non-optimal areas). We then consider the set of unconstrained 2-objective problems from the CEC 2009 competition [31]. The DEB2 problem is defined by:

$$minimise\, G_1(x_1, x_2) = x1 \qquad minimise\, G_2(x_1, x_2) \qquad = \frac{g(x_2)}{x_1} \tag{7}$$

where $g(x_2) = 2.0 - \exp\left(-\left(\frac{x_2-0.2}{0.004}\right)^2\right) - 0.8\exp\left(-\left(\frac{x_2-0.6}{0.4}\right)^2\right)$ and x_1 and x_2 are defined in the range $[0.1, 1.0]$. DEB2 is convenient for preliminary exploration since it is two-dimensional and so suitable for visualisation. It has a local minimum around $x_2 \approx 0.6$ and a global minimum around $x_2 \approx 0.2$, facilitating comparison of the relative behaviour of MOPC with other methods.

In these experiments, all the algorithms used 35,000 evaluations for DEB2 and 300,000 evaluations for the CEC2009 problems, and we present means and standard deviations over 30 independent runs. MOPC used 50 particles, $|A| = 100$, $|D| = 20$, $|L| = 400$, $T_{start} = 1$ and $T_{end} = 0.0001$. NSGAII, MOEA/D-DE and SMPSO implementations where taken from the jMetal framework (`jmetal.sourceforge.net/`v.2.2), all with archive sizes of 100 and other parameters were set as defined in the CEC2009 paper for MOEA/D [30] or as default in JMetal.

4.1 Investigating MOPC on the DEB2 Problem

Measure	MOPC	MOEA/D-DE[15]	NSGAII[8]	SMPSO[21]
HV	0.81956 (0.00317)	0.73897 (0.08384)	0.77543 (0.07794)	0.82331 (0.00003)
IGD	0.05354 (0.04774)	0.15857 (0.13238)	0.09122 (0.12665)	0.01161 (0.00006)
SPREAD	0.43821 (0.15238)	0.87355 (0.14272)	0.44967 (0.08857)	0.10700 (0.01550)

Table 1: Comparison between MOPC, NSGAII and MOEAD on the DEB2 problem

Table 1 gives means and standard deviations of hypervolume, spread and IGD on DEB2. MOPC and SMPSO outperform MOEA/D and NSGAII, approaching the maximum achievable hypervolume (0.82628), while MOEA/D and NSGAII seem prone to fall into the local minima at $x_2 \approx 0.6$. But, SMPSO achieves better spread than MOPC. Recall that MOPC uses the maximin function and hence should prompt a more uniform spread. To explore this, Figure 1 shows the results of single runs of MOPC and NSGAII. MOPC finds the Pareto set in the small region $x_2 \approx 0.2$, and its dynamics can be seen in Figure 2, showing the evolution of the parametric distribution after 1, 10, 20 and 35 (the final) generations.

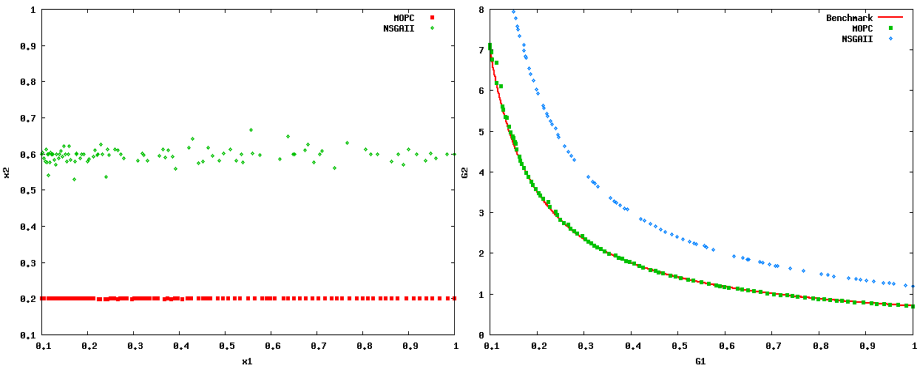

Fig. 1: Decision and objective space for MOPC results on DEB2.

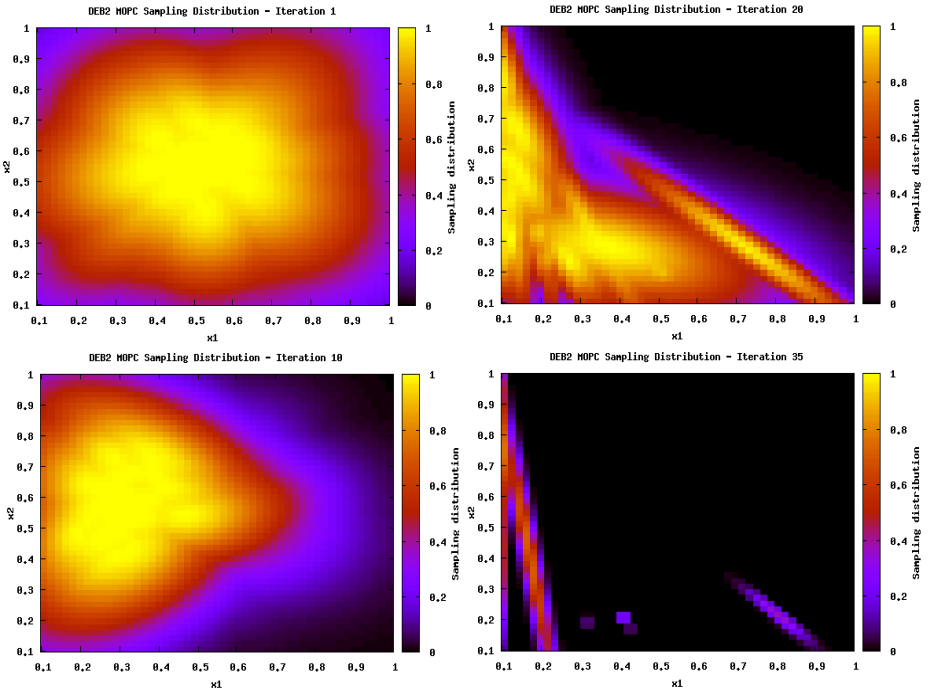

Fig. 2: Parametric distributions of q_θ during optimisation of the DEB2 at iterations 1, 10, 20 and 35.

Initially centred (by definition of the algorithm), from 10 to 20 iterations the distribution slides to the left and down towards the local minima (of weakly non-dominated solutions) where $x_1 = 0.1$ and $x_2 = 0.6$. Over time, it focuses in two areas where x_2 is approx. 0.2 and along the entire length of x_1, which corresponds to the Pareto set, and the area of weakly non-dominated solutions where x_1 is 0.1. This attraction to weakly non-dominated solutions stymies progression of this part of the Pareto set (this can be seen in Figure 1 (b) where the solutions do not exactly match the benchmark set between 0.1 and 0.2).

Overall, the results show that MOPC can find the Pareto set in a small area of the decision space when local minimum exist. Also, MOPC outperforms the MOEA/D and NSGAII implementations tested, but an attraction to weakly non-dominated regions may compromise performance on higher dimensional problems. To investigate this concern, the next section presents experimental results on 30-dimensional problems taken from the CEC 2009 competition [31].

4.2 MOPC Performance on the CEC 2009 problems

Again, our comparative algorithms are SMPSO [21], MOEA/D [15] and NSGA-II [8]; these are respectvely: an exemplar of the high-performing family of MOEAs based on particle swarm optimisation, the (arguably) state of the art MOEA/D in terms of performance (also an exemplar of the approach to MOO of performing

simultaneous single-objective searches in different 'directions', and, finally, the well known accepted benchmark method for MOEAs. They are compared in terms of the IGD metric, which measures the distance from the known Pareto front of the problem in hand. Though not ideal as a comparison metric on MOO [12], this was the chosen metric for the CEC 2009 competition.

Problem	MOPC	MOEA/D-DE[15]	NSGAII[8]	SMPSO[21]
UF1	0.02417 (0.00355)	0.00855 (0.01438)	0.08803 (0.02916)	0.06279 (0.00663)
UF2	0.03857 (0.00147)	0.03397 (0.00038)	0.03831 (0.00102)	0.04393 (0.00157)
UF3	0.17403 (0.01925)	0.01467 (0.01238)	0.15237 (0.03909)	0.12459 (0.03697)
UF4	0.11505 (0.00605)	0.08440 (0.01070)	0.08314 (0.00356)	0.10830 (0.00463)
UF5	0.50165 (0.02940)	0.73351 (0.11537)	0.37613 (0.07276)	0.74538 (0.15191)
UF6	0.11150 (0.01554)	0.11250 (0.05029)	0.12647 (0.05887)	0.33251 (0.03695)
UF7	0.05208 (0.00297)	0.03880 (0.00020)	0.04878 (0.00280)	0.04810 (0.00139)
UF8	0.36911 (0.01215)	0.39651 (0.10624)	0.15329 (0.00682)	0.23546 (0.01406)
UF9	0.15139 (0.00443)	0.39029 (0.03152)	0.17136 (0.01193)	0.18704 (0.02434)
UF10	0.44892 (0.01813)	0.71918 (0.17908)	0.29066 (0.03195)	0.28438 (0.02143)

Table 2: Comparison of IGD for MOPC, MOEAD, NSGAII and SMPSO on CEC2009

The results in Table 2 were produced using the JMetal implementations. We observed that the IGD metric in JMetal was different from that used in the CEC competition, and we note there are several other differences between the jMetal implementation of MOEA/D to the one used in the CEC 2009 competition[30], so, we do not compare our results with those published for the CEC 2009 competition, and consider instead a self-contained comparison among the algorithms described herein.

A broad analysis considering the mean IGD values (lower is better), finds that none of the four algorithms clearly dominates the others. When we consider mean rank of performance over the 10 cases, where 1 is best and 4 is worst, these are respectively 2.2, 2.3, 2.7 and 2.8 for NSGA-II, MOEA/D, MOPC and SMPSO, however this obscures a wide variation in relative performance, Focusing on MOPC, we find its mean is better than MOEA/D in 4 of the 10 cases, better than NSGA-II in 3 of the 10, and better than SMPSO in 5 of the 10. Finally we note there was no attempt to optimise or understand the parameters of MOPC in this initial study.

As a preliminary approach to MOO within the PC framework, it is clear that MOPC indicates some promise for this line of research, showing an ability to outperform, at least in a minority of cases, state of the art MOEAs.

5 Conclusions, Discussion and Future Work

We presented MOPC, a first attempt to design a MOO algorithm within the PC framework. First we explored MOPC on the hard but low-dimensional DEB2

problem, finding that it could approximate the Pareto set well, outperforming NSGAII and MOEA/D, but bettered by SMPSO. We attributed the gap between MOPC and SMPSO to the minimax fitness function, which does not differentiate ideally in the relative selection pressure towards strongly and weakly dominated areas of objective space. On the CEC 2009 problems, MOPC remained promising, outperforming each comparative methods on at least some of the problems. Considering that MOPC is a first approach to MOO within the PC framework, it is clear that its competitive performance against state of the art MOEAs indicates some promise for the PC strategy in multi-objective optimisation.

Regarding future work, it seems plausible that an approach based on decomposition (as in MOEA/D), is likely to perform well within the PC framework. For example, MOEA/D pursues several single objective optimisation in several well-spread 'directions' at once, where a direction corresponds to a specific weighted sum of the original objectives; this can be done in the PC framework by mapping each such direction to a separate PC-based single-objective optimization. A related highly promising direction is the incorporation of local search, whereby PC provides the global search strategy, but each sample is locally optimzed before the distributions are updated. Finally, a further promising thread is to extend the parametric representation used here to a more complex representation (such as mixtures of multivariate Gaussians) to allow a greater diversity of distributions to be modelled by a single particle.

Acknowledgments

Funded by the Systems Eng. for Autonomous Systems (SEAS) Defence Tech. Centre (DTC). Patent pending - UK application no. 0919343.4.

References

1. Bentley, P., Wakefield, J.: Finding acceptable solutions in the Pareto-optimal range using multiobjective genetic algorithms. Springer (1997)
2. Bieniawski, S.: Distributed Optimization and Flight Control Using Collectives. Thesis, Stanford University (2005)
3. Bieniawski, S., Kroo, I., Wolpert, D.: Discrete, continuous, and constrained optimization using collectives. In: Proc. 10th AIAA/ISSMO M.A.O. Conf. NY (2004)
4. Corne, D., Deb, K., Fleming, P., Knowles, J.: The good of the many outweighs the good of the one: Evol. mult. opt. coNNectionS 1(1), 9–13 (2003)
5. Corne, D., Knowles, J.: Techniques for highly multiobjective optimisation: Some nondominated points are better than others. In: Proc. GECCO 2007 (2007)
6. Deb, K.: Multi-objective genetic algorithms: Problem difficulties and construction of test problems. Evolutionary Computation 7, 205–230 (1999)
7. Deb, K.: Multi-Objective Optimization Using EAs. John Wiley and sons Inc (2002)
8. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast elitist multi-objective genetic algorithm: Nsga-ii. IEEE Trans on Evol. Comp. 6, 182–197 (2000)
9. Huang, C., Bieniawski, S., Wolpert, D., Strauss, C.: A comparative study of prob. collectives based multi-agent systems and gas. In: GECCO (June 2005)

10. Jaskiewicz, A.: On the perf. of multiple objective genetic local search on the 0/1 knapsack problem: a comparative exp. IEEE Trans. E.C. 6(4), 402–412 (2002)
11. de Jong, E., Watson, R., Pollack, J.: Reducing bloat and promoting diversity using multi-objective methods. In: GECCO. pp. 11–18. MK (2001)
12. Knowles, J., Corne, D.: On metrics for comparing nondominated sets. In: Proc. 2002 Congress on Evolutionary Computation. pp. 711–716 (2002)
13. Kullback, S., Leibler, R.A.: On information and sufficiency. Ann. Math. Statist. 22(1), 79–86 (1951)
14. Kuo, T., Hwang, S.Y.: Using disruptive selection to maintain diversity in genetic algorithms. Applied Intelligence pp. 257–267
15. Li, H., Zhang, Q.: Multiobjective optimization problems with complicated pareto sets, moea/d and nsgaii. IEEE Trans. Evolutionary Computation (2008)
16. Li, H., Zhang, Q.: Multiobjective Optimization Problems With Complicated Pareto Sets, MOEA/D and NSGA-II. IEEE Trans. Evol. Comp. 13(2), 229–242 (2009)
17. Li, X.: Better spread and convergence: Particle swarm multiobjective optimization using the maximin fitness function. In: Proc. GECCO. pp. 117–128. Springer (2004)
18. Macready, W., Wolpert, D.H.: Distributed optimization. In: International Conference on Complex Systems. Boston, MA (May 2004)
19. Morrison, J., Oppacher, F.: Maintaining genetic diversity in genetic algorithms through co-evolution. In: Advances in AI. pp. 128–138. Springer LNCS 1418 (1998)
20. Mostaghim, S., Teich, J.: Strategies for finding good local guides in multi-objective particle swarm optimization (mopso). In: Proc. of 2008 IEEE Swarm Intelligence Symposium. pp. 26–33. IEEE Service Center (2008)
21. Nebro, A., Durillo, J., García-Nieto, J., Coello Coello, C., Luna, F., Alba, E.: Smpso: A new pso-based metaheuristic for multi-objective optimization. In: 2009 IEEE Symp. on MCDM). pp. 66–73. IEEE (2009)
22. Rajnarayan, D., Wolpert, D.H., , Kroo, I.: Optimization under uncertainty using probability collectives. In: 10th AIAA/ISSMO M.A.O. Conf. (2006)
23. Shore, J., Johnson, R.: Properties of cross-entropy minimization. Information Theory, IEEE Transactions on 27(4), 472–482 (Jul 1981)
24. Sierra, M.R., Coello, C.A.C.: Improving pso-based multi-objective optimization using crowding, mutation and dominance. In: Proc. of EMO 2005. pp. 505–519. Springer LNCS vol. 3410 (2005)
25. Waldock, A., Nicholson, D.: Cooperative decentralised data fusion using probability collectives. In: ATSN-07, at AAMAS-07. pp. 47–54 (2007)
26. Wolpert, D.H.: Information Theory - the bridge connecting bounded rational game theory and statistical physics (2004)
27. Wolpert, D., Strauss, C., Rajnarayan, D.: Advances in distributed optimization using probability collectives. Advances in Complex Systems 9 (2006)
28. Wolpert, D.: Collective intelligence. In: Fogel, D., Robinson, D. (eds.) Computational Intelligence Beyond 2001: Real and Imagined. Wiley (2001)
29. Wolpert, D., Bieniawski, S.: Distributed control by lagrangian steepest descent. In: Proc. 43rd IEEE Conf. on Decision and Control. pp. 1562–1567 (2004)
30. Zhang, Q., Liu, W., Li, H.: The performance of a new version of moea/d on cec09 unconstrained mop test instances. In: CEC'09: Proceedings of the Eleventh conference on Congress on Evolutionary Computation. pp. 203–208. Institute of Electrical and Electronics Engineers Inc., The (2009)
31. Zhang, Q., Zhou, A., Zhao, S., Suganthan, P., Liu, W., Tiwari, S.: MOO test inst. for CEC 09 spec. session and comp. Tech. Rep. CES-887, U. Essex and NTU (2008)
32. Zitzler, E., Deb, K., Thiele, L.: Comparison of multiobjective evolutionary algorithms: Empirical results. Evolutionary Computation 8(2), 173–195 (2000)