Many-Objective Particle Swarm Optimization by Gradual Leader Selection

Mario Köppen and Kaori Yoshida

Kyushu Institute of Technology, Dept. Artificial Intelligence,
680-4, Kawazu, Iizuka, Fukuoka 820-8502 Japan
{mkoeppen, kaori}@pluto.ai.kyutech.ac.jp

Abstract. Many-objective optimization refers to multi-objective optimization problems with a number of objectives considerably larger than two or three. This papers contributes to the use of Particle Swarm Optimization (PSO) for the handling of such many-objective optimization problems. Multi-objective PSO approaches typically rely on the employment of a so-called set of leaders that generalizes the global best particle used in the standard PSO algorithm. The exponentially decreasing probability of finding non-dominated points in search spaces with increasing number of objectives poses a problem for the selection from this set of leaders, and renders multi-objective PSOs easily unusable. Gradual Pareto dominance relation can be used to overcome this problem. The approach will be studied by means of the problem to minimize the Euclidian distances to a number of points, where each distance to the points is considered an independent objective. The Pareto set of this problem is the convex closure of the set of points. The conducted experiments demonstrate the usefulness of the proposed approach and also show the higher resemblance of the proposed PSO variation with the standard PSO.

1 Introduction

Recently, there has been growing interest in the application of particle swarm optimization (PSO) to the handling of multi-objective optimization problems. Since the initial presentation of the MOPSO algorithm (Multi-Objective Particle Swarm Optimization) [2], a growing number of proposals about corresponding standard PSO variations can be found in the literature. The recent survey of Reyes-Sierra and Coello [10] already classifies nearly thirty of such algorithms. According to [10], the general structure of any such PSO variant can be seen as given in Algorithm 1.1 (also covering the standard "single-objective" PSO). The main difference to a PSO is the notion of "leaders," which generalizes the common concept of the global best particle in the standard PSO. This regards the fact that in multi-objective optimization, usually, there is not a single optimum but a set of optima solutions. Without the support of an additional, external "decision maker" instance, the problem statement does not entail any further selection criteria that can be applied to this set of optima.

MOPSO and all its successors proved to be competent algorithms to handle the domain of multi-objective optimization, at least for problems posing two or three conflicting objectives. However, no efforts so far have been devoted to the handling of a notably larger number of objectives. More and more, problems with a larger number of objectives are appearing in practice and deserve a deeper study of the question whether they could be handled by the PSO heuristic as well.

Algorithm 1.1: General Particle Swarm Optimization([10])

```
Initialize swarm
Locate leader
g ← 0
while g < G
do { for each particle
     do { Update position (Flight)
          Evaluation
          Update lbest
   Update leader
   g ← g + 1
```

Notably, approaches to handle these so-called many-objective optimization problems are likely to suffer from the so-called "curse of dimensionality." In this paper, we will stress on this and identify a weak point in the common scheme for selecting among the leaders, related to the rapdidly decreasing probability of finding such leaders in the search space at all. We will propose an approach to overcome this drawback by using a gradual ordering relationship among particles, and compare its performance by means of a scalable many-objective optimization problem that we are going to introduce here as well.

Section 2 will provide the necessary algorithmic concepts and solicit the problem statement. Due to space limitations, multi-objective optimization and PSO will be only briefly touched. The reader is suggested to consider the excellent books of Coello et al.[3] and Deb et al.[4] about multi-objective optimization, and the book of Kennedy and Eberhart to learn about PSO[6]. The used ranking by Fuzzy Pareto Dominance (FPD), which is employed in the PSO generalization, will be recalled in section 2 as well. The so-called P* many-objective optimization problem will be introduced in section 3. Section 4 then provides the results of test runs of multi-objective PSOs on this problem, and a discussion of the results. The concluding section and the references will be at the end of this paper.

2 Multi-Objective Particle Swarm Optimization

In multi-objective optimization, the mapping of a feature vector space F into an objective vector space O is considered, where points of F are seeked, having all their objective values as small as possible. For comparing two points in objective

space, the common notion of Pareto dominance (or just dominance) is employed. Given two vectors $x = (x_1, x_2, \ldots, x_n)$ and $y = (y_1, y_2, \ldots, y_n)$ of same size, it is said that vector x "Pareto dominates" y, if each component of x is smaller than or equal to the corresponding component of y, and at least one component is smaller: $x <_D y \leftrightarrow \forall i(x_i \leq y_i) \wedge \exists k(x_k < y_k)$. A similar definition for the maximum case, or the case "smaller/larger-or-equal" (so-called weak dominance) can be provided as well.

For simplicity, we also consider the dominance relation among points in the feature space, if their corresponding objective vectors are in such a relation. The set of all objective vectors assigned to feasible feature vectors, which are not dominated by any other objective vector is the so-called "Pareto front" of the multi-objective optimization problem. The set of the corresponding feature vectors will be referenced as Pareto set in the following.

The general task of an multi-objective optimization algorithm is to find a representation of the Pareto front of the given mapping. A new class of evolutionary algorithms, with specialized selection operators to regard for the multiple objectives, has gained much attractiveness for the handling of such problems. As already said in the introduction section, this interest has expanded to the Particle Swarm Optimization algorithm in between.

Here, we consider the multi-objective PSO (MOPSO) that was recently proposed by Alvarez et al.[1], see the algorithm listing 2.1. It shows the same global structure as the general PSO given in listing 1.1. The procedure to select the leader is based on selecting from an archive that stores all non-dominated positions found by the algorithm during its course so far.

Algorithm 2.1: MOPSO$_{rand}$([1])

```
A ← ∅
Initialize particles
for g ← 1 to G
do { for n ← 1 to N
     do { for k ← 1 to K
          do { v_nk = w v_nk + r1(P_nk − x_nk) + r2(G_nk − x_nk)
               x_nk = x_nk + v_nk + ε
          x_n ← enforceConstraints(x_n)
          o_n ← f(x_n)
          if ∼ ∃u ∈ A : u ≤_D x_n
            then { A ← {u ∈ A | x_n ≰_D u}
                   A ← A ∪ x_n
          if x_n ≤_D P_n ∨ (x_n ≮_D P_n ∧ P_n ≮_D x_n)
            then P_n ← x_n
          G_n ← selectGuide(x_n, A)
```

The parameters used in the listing are: x_n and v_n are position and velocity of the n-th particle, A stands for the archive, G is the maximum number of cycles, N is the number of particles, K the dimension of a particles' vector, P_n is the local best position of particle x_n, and G_n the leader (or guide) for this particle, used to update its position according to the general swarm

equations. The parameters r_1 and r_2 are random numbers drawn from the interval $[0, \ldots, w_{local}]$ and $[0, \ldots, w_{global}]$ respectively, w is the weight of former velocity, and ϵ so-called curl parameter, a small random number. The procedure $enforceConstraints()$ ensures the swarm to stay within the feasible space, and the procedure $selectGuide()$ selects the leader for each particle, and will be detailed in a moment.

2.1 Choices for the leader selection

In [1], three procedures for the selection of the leader from the archive were proposed. All of them assume the presence of particles being dominated by vectors that are already stored in the archive, or being dominated by new positions. The procedure $RAND$ randomly selects one of the archive members dominating particle x_n as leader for this particle. If there is no such particle, an archive member is randomly selected.

However, for a larger number of objectives, the second choice becomes more likely. Equation (1), taken from [7], gives the expected size of the Pareto front of m points randomly selected from the n-dimensional unit hypercube.

$$e_m(n) = m - \sum_{k=1}^{m} \frac{(-1)^{k+1}}{k^{n-1}} \binom{m}{k} \tag{1}$$

A closer look on this equation gives that the probability of finding two points with one dominating the other drops exponentially with the problem dimension. For example, for 15 objectives and 10 particles, the probability to have a randomly selected dominated point is already as low as 0.0027.

For this reason, we are considering gradual Pareto dominance here, as it was presented in [9]. Given a set of points, a "ranking value" is assigned to each point a. This ranking value is the maximal value of the "degree of being dominated" by any other element of the set. The dominance degree is computed as

$$\mu_{pmin}(x, y) = \prod_i \left[\frac{x_i}{y_i}\right] \tag{2}$$

where the notion of a bounded division was used:

$$\left[\frac{x}{y}\right] = \begin{cases} 1, & \text{if } y \leq x \\ x/y, & \text{if } x < y \end{cases} \tag{3}$$

The ranking value of a dominated point is 1, otherwise its between 0 and 1. Points with smaller ranking values can be considered to be less dominated by the other points in the set. The ranking value is scale independent, and points close to other points in the set yields higher ranking values than points that are more distant. Thus, the ranking values also punish crowding of points at the same location. Finally, the ranking value is set-dependent. If in a set a point x has a lower ranking value than y, adding an element to the set close to x may reverse the ranking value size relation.

In this paper, the procedure FPD for *selectGuide*() selects as leader for all particles the particle with the lowest ranking value within the set of all swarm particles.

3 The P* Many-Objective Optimization Problem

The evolutionary multi-objective optimization community maintains a set of benchmark measures for the performance assesment of algorithms, with the DTLZ suite of problems [5] being among the most popular. Unfortunately, not much is known about these problems for the many-objective case. This circumstance has already been remarked with the introduction of the Pareto Box problem, which identifies objective vector and feature vector [8]. However, issues of vector components becoming 0 and the bounded domain of positive vector components hardens the use of the Pareto Box problems, especially for swarms.

Here, we are introducing a related problem, referred to as *P* problem* for indicating the variable number of points from which the objectives are derived.

Given is a set P of m points P_i in the Euclidian plane (the case of two dimensional Euclidian space is completely sufficient for the present analysis). The feature space F equals the Euclidian plane, where the points P_i are located. The objective space O is an m-dimensional vector space. For a given point x in the feature space, its objective vector $o(x)$ is the vector with the components $o_i = d(x, P_i)$ for $i = 1$ to m, where $d(x, y)$ is the Euclidian distance of two points $x, y \in F$. Thus, the objectives to minimize are the distances to a given collection of points, where the distance to any of these point is treated as an independent objective.

The Pareto set of this problem equals the convex closure of the points P_i. To see this, consider fig. 1. The left subfigure shows that for any point x outside the convex closure there is at least one point y that is closer to all points of P. In the subfigure, the line $\overline{A_i A_{i+1}}$ represents one segment of the convex hull. All points on this line or on the other side of this line than x are more close to y than to x.

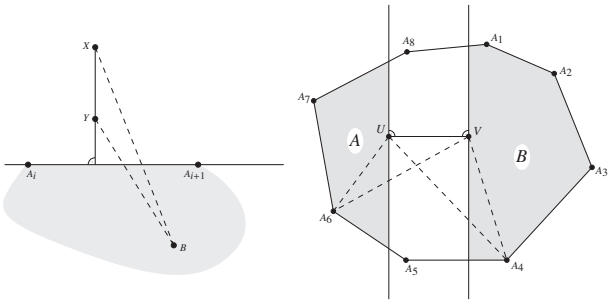

Fig. 1. Proof that the Pareto set of the P* problem for some points A_i is the convex closure of these points.

To see that none of the points of the convex closure dominates any other, consider the right subfigure of fig. 1. By connecting any two points U and V of the convex closure and drawing the perpendiculars to this line trough U and through V, the convex closure is segmented into three parts. There is at least one point of the point set A (and thus of P) located to the l.h.s. of the perpendicular through U (indicated by encircled A in the figure), or located on this line, and there is at least one point of A belonging to the r.h.s. of the perpendicular through V or on it (indicated by encircled B). Otherwise, the shape would not be convex. Now, point U is more close to any point of A than V, and point V is more close to any point of B than U. Neither U nor V can dominate the other.

Having thus a rather simple solution structure, the problem is worth a study for a heuristic algorithm for several reasons:

- the number of objectives can be easily scaled
- by reducing the area enclosed by the convex closure, the effort for random search (the "Monte-Carlo Barrier") can be easily increased
- typical performance measures (as average distance to Pareto front, number of particles belonging to the Pareto front) can be directly computed
- as the feature space is two-dimensional, the results can be directly visualized; however, extension to higher-dimensional spaces is straightforward
- the search space is not bounded
- the problem is a continuous optimization problem
- boundary conditions can be directly included
- crowding in objective space directly corresponds to crowding in feature space
- modelling of algorithm behaviour seems feasible
- by using the distance to the center of gravity of the points instead, a comparison to the single-objective case becomes possible

In the following, we are considering the performance of three algorithms on the P* problem.

4 Results and Discussion

Several experiments have been conducted to study the behaviour of three algorithms on the P* problem. The three algorithms were the MOPSO of Alvarez et al. using $RAND$ (algorithm $MOPSO_{rand}$), the proposed usage of FPD (algorithm $MOPSO_{fpd}$) as procedures for $selectGuide()$, and the standard PSO (algorithm PSO) by taking the distance to the c.o.g. of the points P as objective. The result that we want to present here was achieved with the following settings: swarm sizes were 10 particles each; the swarms were randomly initialized around point 0, with deviation of 0.1 and max initial velocity of 0.001; the weight of the global best was 0.08, the weight of the local best 0.02, the weight of the former velocity 0.985. For each target problem, the average distance of the c.o.g. of the particle positions after 1000 cycles for 20 different random initializations was computed. The target problems were given by placing a set of 15 circular points at a radius of 0.01 around the center $(i/10, i/10)$ with i going from 1 to 20.

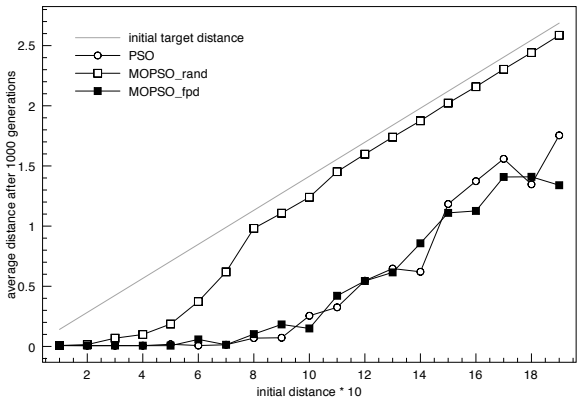

Fig. 2. Plots of the average distances to the target after 1000 cycles vs. different initial target distances for a problem with 15 objectives.

The results can be seen in fig. 2. For reference, the initial distance of the target to the starting point of the swarm has been plotted as well. The most notable fact is the nearly complete failure of the algorithm $MOPSO_{rand}$ to find the target set, once the target gets placed beyond the $(0.8, 0.8)$ offset. The constant bias of about 0.1 of $MOPSO_{rand}$ performance to the reference line refers to the fact that the initial positions varied by about 0.1 around the point 0. Before the failure, it has to be noted that the averages for $MOPSO_{rand}$ are taken from more or less binary cases: the $MOPSO_{rand}$ swarm was either reaching the target set, or got stuck at the initial position.

For understanding this failure, consider a snapshot of the $MOPSO_{rand}$ swarm particles taken anywhere in such a situation (see fig. 3). The snapshot also shows the positions of the local best, as they are kept by each individual, and the target points. It can be seen that in such a situation, the local best positions establish a kind of "trap" for the swarm. The swarm is surrounded by these positions, and the probability of finding a closeby position that either dominates the local best positions, or can get added to the archive is nearly zero. The archive so far is not able to guide the swarm out of this trap. So, the swarm stays within the area surrounded by the local best positions, and the local best positions never get updated by new dominating positions. Thats the reasoning, given in humble words. Any manner of quantifiying the situation of a MOPSO getting stuck will only provide additional evidence for the fact that this is a ubiquitous feature of all algorithms that depends on the finding of dominated points, to perform their operations.

The algorithm $MOPSO_{fpd}$ shows a much more improved performance, as it is capable to appraoch the target even if being initially placed distant from the target set. However, naturally the explorational effort for $MOPSO_{fpd}$ is also increasing. But it has to be taken into account that the average values shown for

the $MOPSO_{fpd}$ were taken from a variety of distance values: in nearly no case, the $MOPSO_{fpd}$ swarm got ever stuck at the initial position, as it happened many times for the $MOPSO_{rand}$. It could always escape the trap set up by the local best, even if no dominating position was found, as it is also rewarding nearly dominating positions. By extending the number of cycles, the $MOPSO_{fpd}$ swarm may still approach the target.

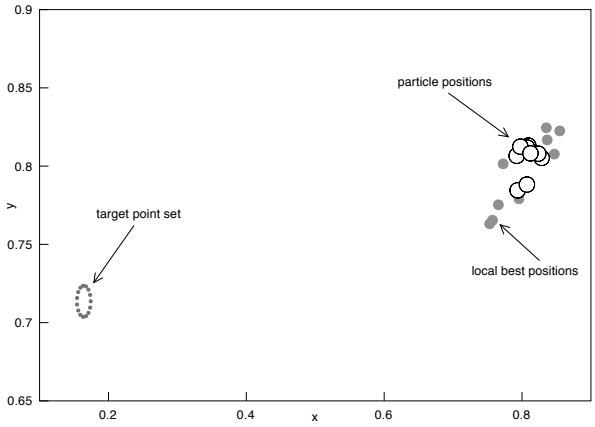

Fig. 3. A situation where a $MOPSO_{rand}$ swarm is trapped within the local best positions of the particles.

The performance measure for the standard PSO has been taken as well. It is remarkable how strongly $MOPSO_{fpd}$ performance resembles the one of the standard PSO, as both plots go nearly equal. This encourages the choice of the FPD-based leader selection scheme as a way to expand the standard PSO to the many-objective optimization domain.

5 Conclusions

In this paper, the extension of Multi-Objective Particle Swarm Optimization (MOPSO) to the case of many-objective optimization has been studied. Here, "many-objective" stands for a number of objectives considerably larger than two or three. It has been demonstrated how algorithms that rely on the presence of dominated points may get stuck in their search for the Pareto front. The notion of Pareto dominance is not fully suited to the case of many objectives. Pareto dominance requires ALL components of a vector to be smaller than the corresponding components of the other vector. This becomes more and more unlikely, as the number of components increases. As a consequence, such an MOPSO algorithm may get trapped by the local best selection of the particles itself, without being able to find new dominating positions. As a countermeasure,

this paper presented the use of gradual Pareto dominance, to fuse the relative amount of smaller vector components, and the degree by which they are smaller, into a single measure. Doing leader selection based on these so-called "ranking values" allows for the design of a MOPSO, which better resembles a standard single-objective PSO in the multi-objective case.

Acknowledgment

A researcher involved in this study has been supported by a JSPS grant.

References

1. Julio E. Alvarez-Benitez, Richard M. Everson, and Jonathan E. Fieldsend. A MOPSO Algorithm Based Exclusively on Pareto Dominance Concepts. In Carlos A. Coello Coello, Arturo Hernández Aguirre, and Eckart Zitzler, editors, *Evolutionary Multi-Criterion Optimization. Third International Conference, EMO 2005*, pages 459–473, Guanajuato, México, March 2005. Springer. Lecture Notes in Computer Science Vol. 3410.
2. Carlos A. Coello Coello and Maximino Salazar Lechuga. MOPSO: A Proposal for Multiple Objective Particle Swarm Optimization. In *Congress on Evolutionary Computation (CEC'2002)*, volume 2, pages 1051–1056, Piscataway, New Jersey, May 2002. IEEE Service Center.
3. Carlos A. Coello Coello, David A. Van Veldhuizen, and Gary B. Lamont. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Kluwer Academic Publishers, New York, May 2002. ISBN 0-3064-6762-3.
4. Kalyanmoy Deb. *Multi-Objective Optimization using Evolutionary Algorithms*. John Wiley & Sons, Chichester, UK, 2001. ISBN 0-471-87339-X.
5. Kalyanmoy Deb, Lothar Thiele, Marco Laumanns, and Eckart Zitzler. Scalable Multi-Objective Optimization Test Problems. In *Congress on Evolutionary Computation (CEC'2002)*, volume 1, pages 825–830, Piscataway, New Jersey, May 2002. IEEE Service Center.
6. James Kennedy and Russell C. Eberhart. *Swarm Intelligence*. Morgan Kaufmann Publishers, San Francisco, California, 2001.
7. Mario Köppen, Raul Vicente Garcia, and Bertram Nickolay. Fuzzy-pareto-dominance and its application in evolutionary multi-objective optimization. In *Evolutionary Multi-Criterion Optimization, Third International Conference, EMO 2005, Guanajuato, Mexico, March 9-11, 2005. Proceedings*, LNCS 3410, pages 399–412. Springer Berlin / Heidelberg, 2005.
8. Mario Köppen, Raul Vicente Garcia, and Bertram Nickolay. The pareto-box problem for the modelling of evolutionary multi-objective optimization. In *Adaptive and Natural Computing Algorithms. Proceedings of the ICANNGA 2005, Coimbra, Portugal*, pages 194–197, 2005.
9. Mario Köppen and Raul Vicente Garcia. A fuzzy scheme for the ranking of multivariate data and its application. In *Proceedings of the 2004 Annual Meeting of the NAFIPS (CD-ROM)*, pages 140–145, Banff, Alberta, Canada, 2004. NAFIPS.
10. Margarita Reyes Sierra and Carlos A. Coello Coello. Multi-objective particle swarm optimizers: A survey of the state-of-the-art. *International Journal of Computational Intelligence Research*, 2(3):287–308, 2006.