Multi-Objective Particle Swarm Optimization by Fuzzy-Pareto-Dominance Meta-Heuristic

Mario Köppen[1], Christian Veenhuis[2]
[1]Kyushu Institute of Technology, Dept. Artificial Intelligence,
680-4, Kawazu, Iizuka, Fukuoka 820-8502, Japan
[2]Fraunhofer IPK, Dept. Security Technology
Pascalstr. 8-9, 10587 Berlin, Germany
E-mail:mkoeppen@ieee.org,veenhuis@ipk.fraunhofer.de

Abstract

This paper introduces a new approach to multi-objective Particle Swarm Optimization (PSO). The approach is based on the recently proposed Fuzzy-Pareto-Dominance (FPD) relation. FPD is a generic ranking scheme, where ranking values are mapped to element vectors of a set. These ranking values are directly computed from the element vectors of the set and can be used to perform rank operations (e.g. selecting the "largest") with the vectors within the given set. FPD can be seen as a paradigm or meta-heuristic to formally expand single-objective optimization algorithms to multi-objective optimization algorithms, as long as such vector-sets can be defined. This was already shown for the Standard Genetic Algorithm. Here, we explore the application of this concept to PSO, where a swarm of particles is maintained. The resulting PSO$_{f2r}$ algorithm is studied on a fundamental optimization problem (so-called Pareto-Box-Problem) where a complete analysis is possible. The PSO$_{f2r}$ algorithm is shown to handle the case of a larger number of objectives, and shows similar properties like the (single-objective) PSO.

1 Introduction

Multi-objective Optimization is still a challenging and stimulating research and application field for Evolutionary Computation. A multitude of approaches and discussions on related topics have been provided. Formulating the major tasks of multi-objective optimization as 1.) find the Pareto front of a given multi-objective optimization problem; 2.) diversify on the Pareto front; and 3.) qualify the parts of the Pareto front for supporting the computational instance of a so-called *decision maker*; EC has been shown to supply a rich and versatile portfolio of methods to attach to all these tasks.

Recently, the multivariate ranking scheme Fuzzy-Pareto-Dominance (FPD) was introduced [15]. FPD maps a set of n vectors into the $[0, 1]^n$ range, to so-called *ranking values*, fulfilling the condition that all dominated vectors are mapped to 1 and all non-dominated vectors are mapped to values smaller than 1. The smaller the assigned ranking value, the lower the *degree* of this vector, to which it is dominated by any other vector of this set.

The procedure is *set-based*, i.e. the ranking value of a vector depends on the set, to which it belongs. Adding or removing other elements to or from this set may influence the numerical value of the ranking value of that vector. However, given a particular set, the ranking values can be used to apply any ranking- or sorting-related procedure on the vectors. Thus, it can be used in population-based algorithms like in evolutionary computation, to extend the notion of rank to the multi-objective case. In [14] this was shown for the case of the Standard Genetic Algorithm (SGA). In each generation step, the vector-objectives of the population are replaced by the ranking values of these vector-objectives in the set of the individuals. Then, the ranking values are used like fitness values in the genetic operators, i.e. to base roulette-wheel selection or tournament selection on them.

It was suspected that the "replace vector-objectives by ranking values" is a more general paradigm that can be used to make a single-objective population-based optimization algorithm capable to handle the multi-objective case as well. The claim came out to be correct for the SGA case. But how is the situation for other popular algorithms in this field?

There is a 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) [6], a growing number of proposals about corresponding standard PSO variations can be found in the literature (see for example [18] [10] [20] [25] [17] [7] [3] [2] [26] [16] [11] [19] [4] [8] [9] [22] [21]). The recent survey of Reyes-Sierra and Coello [24] already classifies nearly thirty of such algorithms. 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.

In this paper, we are going to investigate the application of the above mentioned paradigm to the case of Particle Swarm Optimization (PSO). Here, we are considering a rather canonical way of doing such an extension, by keeping the single-objective design of the PSO itself unmodified. In section 2, the major prerequisites for this approach are recalled: the PSO and the FPD equations.

Then, section 3 introduces the multi-objective version of PSO, achieved by incorporating the corresponding selection operations (global and local best) on the FPD ranking values. Then, in section 4 the resulting PSO$_{f2r}$ algorithm is verified by means of the Pareto-Box-Problem. The paper ends with conclusions and reference section.

2 Prerequisites

In this section, we recall the major ingredients that are needed to establish a multi-objective version of the PSO algorithm: the Fuzzy-Pareto-Dominance paradigm, and the Particle Swarm Optimization.

2.1 Fuzzification of Pareto Dominance Relation

In this section, we are going to study the fuzzification of the Pareto dominance relation. For two vectors a and b it is said that a *(Pareto-)dominates* b, if each component of a is less or equal to the corresponding component of b, and at least one component is smaller:

$$a \ >_D \ b \longleftrightarrow \forall i(a_i \leq b_i) \wedge \exists k(a_k < b_k). \tag{1}$$

Note that in a similar manner Pareto dominance can be related to the $>$-relation, depending on the application context.

The subset of all vectors of a set M of vectors, which are not dominated by any other vector of M is the Pareto set (also Pareto front). The Pareto set for univariate data (single objective) contains just the maximum of the data.

The goal of the fuzzification of this concept is to yield a "softer" and practically usable numerical representation of the dominance relation between two vectors that can be employed in EMO. The issue was studied in more detail in [15]. This work showed the principal problems related to the specification of such a degree of dominance. Fuzzy dominance degrees can be computed once the following two conditions are taken into account:

1. The measure is not symmetric, and between two vectors a and b the two measures "a dominates b by degree α" and "a is dominated by b to degree α" have to be distinguished. Moreover, if a dominates b, either one measure is numerically 0 and the other lower-or-equal to 1, or one is greater-or-equal to 0 and the other 1[1].

2. The dominance degrees are set-dependent and can not be assigned in an absolute manner to single vectors alone. We will refer to the set used for the following computations as the *ranking set*.

[1]In [15] it was demonstrated that otherwise the complexity of the corresponding function specification would grow exponentially, as well as the number of its discontinuities.

A generic fuzzy ranking scheme for a set S of multivariate data (vectors) a_i with real-valued components a_{ij} and $1 \leq i \leq N$ can be based on the provision of a *comparison function* $f_x(y) : \mathbb{R} \times \mathbb{R} \rightarrow [0, 1]$ and a T-norm. Then, the following two steps are performed:

1. We compute the *comparison values* for any two vectors $a_i = (a_{ik})$ and $a_j = (a_{jk})$ by $c_{a_i}(a_j) = T(f_{a_{ik}}(a_{jk}) \,|\, k = 1, \ldots, M)$ with M the number of components of each vector.

2. We compute the *ranking values* for any element a_i of S by

$$r_S(a_i) = \max[c_{a_i}(a_j) | j \neq i].$$

Then, we consider vectors with lower numerical ranking values to be on a higher ranking position.

When using the comparison function bounded division and the algebraic (or product) norm as T-norm, the ranking scheme fulfills several useful properties like scale-independency in the data. The fuzzification of Pareto dominance relation can be written then as follows: It is said that vector *a dominates* vector b by degree μ_a with

$$\mu_a(a, b) = \frac{\prod_i \min(a_i, b_i)}{\prod_i a_i} \tag{2}$$

and that vector *a is dominated by* vector b at degree μ_p with

$$\mu_p(a, b) = \frac{\prod_i \min(a_i, b_i)}{\prod_i b_i} \tag{3}$$

Note that the subscript a stands for so-to-say "active" and p "passive" dominance. For a Pareto-dominating b, $\mu_a(a, b) = 1$ and $\mu_p(b, a) = 1$, but $\mu_p(a, b) < 1$ and $\mu_a(b, a) < 1$. Note that the case of having an a_i or b_i equal to 0 is handled by the exclusion of the corresponding index in the products in the numerator and denominator.

2.2 Particle Swarm Optimization

Particle Swarm Optimization (PSO), as introduced by Kennedy and Eberhart [12] [13], is an optimization algorithm based on swarm theory. The main idea is to model the flocking of birds flying around a peak in a landscape.

In PSO the birds are substituted by particles and the peak in the landscape is the peak of a fitness function. The particles are moving through the search space forming flocks around peaks of fitness functions.

Let N_{dim} be the dimension of the problem (i.e., the dimension of the search space $\mathbb{R}^{N_{dim}}$), N_{part} the number of particles and $\mathcal{P}$ the set of particles $\mathcal{P} = \{P_1, ..., P_{N_{part}}\}$. Each particle $P_i = (x_i, v_i, l_i)$ has a current position in the search space ($x_i \in \mathbb{R}^{N_{dim}}$), a velocity ($v_i \in \mathbb{R}^{N_{dim}}$) and the locally best found position in history, i.e., the own experience ($l_i \in \mathbb{R}^{N_{dim}}$) of this particle.

In PSO, the set of particles $\mathcal{P}$ is initialized at time step $t = 0$ with randomly created particles $P_i^{(0)}$. The initial l_i are set to the corresponding initial x_i. Then, for any time step t, the next position $x_i^{(t+1)}$ and velocity $v_i^{(t+1)}$ of each particle $P_i^{(t)}$ is computed as shown in eqns. (4) and (5).

$$\begin{aligned} v_i^{(t+1)} &= w_I^{(t)} v_i^{(t)} + w_L R_{0,1}(l_i^{(t)} - x_i^{(t)}) + \\ &\quad + w_N R_{0,1}(n_i^{(t)} - x_i^{(t)}) \qquad (4) \\ x_i^{(t+1)} &= x_i^{(t)} + v_i^{(t+1)} \qquad (5) \end{aligned}$$

Here, $R_{0,1}$ means a random number from $[0, 1]$. $n_i^{(t)} \in \mathbb{R}^{N_{dim}}$ represents the best found local position of the best neighbor particle at time t. Because there are several possibilities to define the neighborhood of a particle [13], the best neighboring particle can be, e.g., the best particle of a pre-defined neighborhood; the best particle of the nearest neighbors according to the distance in search space; the globally best particle etc. The inertia weight $w_I^{(t)}$ determines the influence of the particle's own velocity, i.e., it represents the confidence of the particle to its own position (typically $w_I \in [0.1, 1.0]$). To yield a better convergence, this weight is decreased over time [23] [13]. w_L is the influence of the best local position found so far. The influence of the best particle of the neighborhood is denoted with w_N.

To avoid chaotic behavior, the new velocity $v_i^{(t+1)}$ is clamped to a pre-defined interval $[-V_{max}, +V_{max}]$.

The fitness of a particle is determined by a fitness function $F : \mathbb{R}^{N_{dim}} \rightarrow \mathbb{R}$. If the new position $x_i^{(t+1)}$ has a better fitness than the best solution found so far for particle P_i, it is stored in memory as shown in eq. (6) (in case of minimization).

$$l_i^{(t+1)} = \begin{cases} x_i^{(t+1)} & , \quad F(x_i^{(t+1)}) < F(l_i^{(t)}) \\ l_i^{(t)} & , \quad otherwise \end{cases} \qquad (6)$$

The best solution of the run is found at particle P_b with the best local solution l_b. Best solution l_b is always element of the set of all best local solutions $\{l_i\}, \forall i \in \{1, \cdots, N_{part}\}$. The best fitness value is $F(l_b) = \min_{i \in \{1,\cdots,N_{part}\}}\{F(l_i)\}$.

3 Design of the Multi-Objective PSO

As stated in the foregoing, the basic goal is to replace evaluations based on the ranking of fitness values by corresponding ranking values in suitable sets. In PSO, we can identify two such ranking based evaluations: the position of the global best and the position of the local best.

Selection of the Global Best: The first task is rather straightforward to solve. We select all particle objective vectors as the ranking set, compute the ranking values of the objectives of all particles in the swarm and select the one with the lowest ranking value: this is the particle whose objective vector *is dominated by* other particle objective vectors together to the lowest degree (i.e. in the sense of "passive" dominance, using eq. (3)).

Selection of the Local Best: The second task is not straightforward regarding the choice of the ranking set. In single-objective PSO, the local best is usually a former position of the particle, having the best fitness value achieved ever. Using the archive of all former particle positions (i.e. the PARETO set of all former objective vectors) offers two problems:

1. The computational effort will increase with the number of steps.
2. The fact that non-dominated points are usually in a neighborhood of other non-dominated points can trap the swarm at some local regions of the PARETO front.

To resolve these issues, it was considered to store only a fixed number N_a of former objectives in the archive in a queue-like manner (first-in-first-out) and select the position with the lowest ranking value as local best.

Curl Parameter: Also related to the second problem (crowding of individuals at local regions of the PARETO front) it was advisable to provide a means to force diversification at the PARETO front. This was achieved by the common addition of a random term to eq. (3). To each component of the velocity, a random number from $[-t, t]$ was added, with t being the so-called curl parameter (it will be seen later that t causes the swarm to leave straight paths through lines of the search space composed of non-dominated points).

Otherwise, the processing for the PSO was not changed for achieving the FPD-driven algorithm PSO_{f2r} (the subscript $f2r$ annotates the use of *ranking values* instead of *fitness values*).

4 Evaluation for the Pareto-Box-Problem

The performance of EMO algorithms is not much known for the case of an increasing number of objectives, and there is a lack of well-studied benchmark problems . So far, most of the presented approaches are using two or three objectives. For more objectives, EMO creators are in good hope that there is no essential need to change anything on the algorithms for using them in the context of a larger number of objectives (say 5 to 20). However, a small study on the, seemingly very simple, Pareto-Box-Problem [14] that equals property space and objective space (objective function $y_i = f_i(x) = x_i$) shows that algorithms that explicitly refer to the presence of PARETO dominance cases among the individual objective vectors (like the computation of PARETO strength) get in trouble to achieve their goal (to direct a genetic operator) due to the rapidly falling probability to have such a PARETO dominance case at all.

The nD-Pareto-Box-Problem is stated as follows. Given are m uniformly randomly selected n-dimensional points P_i in the n-dimensional unit hypercube ($1 \leq i \leq m$), with coordinates P_{ij} ($1 \leq j \leq n$). Thus, for each P_{ij} we have $0 \leq P_{ij} \leq 1$. The problem we state is:

Pareto-Box-Problem: *What is the expectation value for the size of the* PARETO *set of these points?*

Obviously, the PARETO set of this problem is not hard to find (it only contains the point 0), and there is also no conflict in the objectives. An analysis of the Pareto-Box-Problem [14] gives the expectation value of the size of the PARETO set of m randomly selected values in the n-dimensional unit hyper-cube as

$$e_m(n) = \sum_{k=1}^{m} \frac{(-1)^{k+1}}{k^{n-1}} \binom{m}{k} \tag{7}$$

Taking e.g. $n = 20$ this gives for $m = 1000$ randomly selected points in the hyper-cube an expectation value for the size of the PARETO set of these 1000 points as 999.135. Means roughly only one PARETO dominance case can be expected in a population of 1000 individuals at all. Obviously, this does not suffice to base the computations for a genetic operator on it. The expression for $e_m(n)$ rapidly approaches m (i.e. only the contribution to the sum for $k = 1$ differs remarkably from 0 for larger n). This means that for larger number of objectives the PARETO set of m random vectors is equal to the set of the vectors itself: there is not a single vector dominated by any other vector in the set. It comes out that pure reliance on the presence of dominance cases, posing no major problem in the two- or three-dimensional case, is misleading for a number of objectives even as 'low'' as ten or so. The *search effort* had to grow exponentially to ensure a sufficiently large number of dominance cases in the population.

However, the Fuzzy-Pareto-Dominance concept was already shown [14] to provide a means to overcome this problem. Literally spoken, the gradual dominance relation allows for also taking cases of "close" PARETO dominance into account, providing e.g. the required selection pressure that the sparse dominance cases can not provide alone.

Here, we study the FPD as meta-heuristic applied to the PSO and its performance on the Pareto-Box-Problem for increasing number of objectives. In the following, the parameter settings for PSO$_{f2r}$, after some experiments, were chosen as follows:

- **Population size** N_{part} was set to 7. It came out that this was sufficient to solve the studied tasks.
- **Inertia weight** w_I was not experimented with and set to 1.0.
- **Weight of the Global Best** w_N was set to 0.01.
- **Weight of the Local Best** w_L was set to 0.03.
- **Maximum Velocity** V_{max} was set to 0.04.
- **Curl** t was set to 0.005. Higher values are not advisable in general.

- **Local Archive Size** N_a for selecting the Local Best was set to 10. Any major influence of this parameter on the performance could not be observed so far.
- **Random Creation Method** for the initial swarm was chosen as uniform.
- **Bounding scheme** for handling particles that leave the hyper-cube was selected as clamping the position values by the hyper-cube bounds.

Note that a stronger influence of the ratio between global and local best weights was observed in the preliminary experiments, as it is known from PSOs in general [5].

Figure 1 shows that the PSO$_{f2r}$ algorithm has no problem to approach the point 0 in the case of lower number of objectives (here $n = 4$). For larger numbers, the search slows down due to a phenomena that can be easily seen and be understood as a kind of relative PARETO fronts. Figure 2 shows the decay of the average distances of all particles in each step from the point 0. This figure also shows the distance values for another multi-objective PSO (Alvarez et al. [1]), using the same parameters. In the PSO of Alvarez et al., the swarm stores all non-dominated particle positions in an external archive. During swarm update, each particle randomly selects its "own" leader among the archive members, by which it is dominated. If there is no dominating member in the archive at all, the leader is randomly selected from the whole archive. This gives the counterpart for the global best position in the PSO algorithm. For the local best position, the algorithm just updates its own local best position, if the new position is dominating the former local best position in objective space. The plot in the figure clearly shows that for 15 objectives, the algorithm is performing a random search only. The swarm keeps an average distance of about 2 from the origin, which is nearly equal to the expectation value of a randomly selected point in the unit hyper-cube. This is directly caused by the sparseness of PARETO dominance, as it was already discussed. Describing the algorithm in terms of sparse PARETO dominance reads like: the global best is randomly selected from any position ever visited by the swarm before, and the local best is always the first position of the particle and never gets updated.
Obviously, in contrary, the PSO$_{f2r}$ algorithm is able to approach the origin. Three things can be observed here that comes out to be typical for the application of PSO$_{f2r}$.

1. In the global scheme, the algorithm approaches the point 0 more or less steadily.
2. There are some plateaus, where the PSO$_{f2r}$ gets stuck for a number of steps before continuing the descent to the origin.
3. The swarm does not come to a halt at the end. This is due to the continued internal movements of the swarm, even when its being close to 0, a consequence of keeping the inertia weight constant. It is also a little bit caused by the clamping of the new position values at 0, since the optimum is located at the border of the search space.

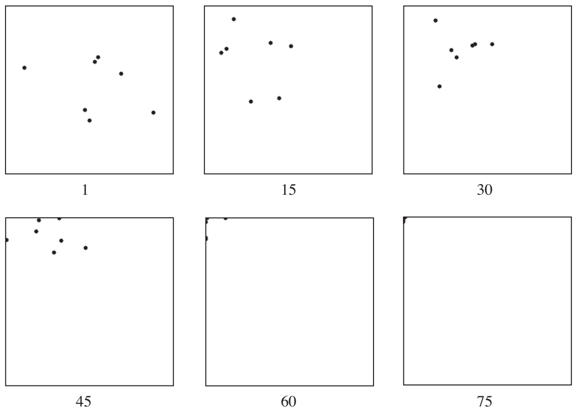

Figure 1: Iterated PSO$_{f2r}$ steps for the 4D-Pareto-Box-Problem. Particle positions are displayed as $(x, y) = (\sqrt{x_1^2 + x_2^2}, \sqrt{x_3^2 + x_4^2})$.

Surely, item 2 can be seen as a counterpart to local optima in the case of single-objective optimization, but here it is definitely algorithm-dependent. Such "relative" Pareto fronts, i.e. search-space locations where the swarm can be entrapped for a while, are related to cases where the objective vector (which equals the position in the Pareto-Box-Problem) approaches a vector that has only components from $\{0, 1\}$. This can be better seen in fig. 3 that shows the probability distribution of swarm particle positions, taken from 30 independent runs of PSO$_{f2r}$ over 500 steps for the 10D-Pareto-Box-Problem. The position vectors are displayed as balanced coordinates:

$$(x, y) = \left(\sqrt{\sum_{i=1}^{5} x_i^2}, \sqrt{\sum_{i=6}^{10} x_i^2} \right)$$

The distribution shows a grid-like pattern, where the PSO$_{f2r}$ spends more time on the nodes of the grid, and is also more often moving along the grid-lines. Despite of the non-unique mapping of vectors from $\mathbb{R}_{15}$ to $\mathbb{R}_2$ involved, this grid is established from position vectors with some of its components, but not all, being 0 (i.e. already optimal). A further analysis gives that these relative Pareto fronts are subsets of the objective space like (in case $n = 2$) $(t, 1 - t)$ with $t \in [0, 0.5]$ where no point dominates any other, but the ranking values are increasing for smaller values of t. The dynamics of the swarm gives that the swarm might start to approach the point $(0, 1)$ instead of $(0, 0)$, and only the random influence quantified by the curl parameter gives the means to escape from this doomed track. Without this curl, the swarm might become unable to leave such relative Pareto fronts.

In the case of increasing dimension, this mechanism seems to become more and more unable to avoid the entrapment in relative Pareto fronts, as table 1 indicates. Shown is the average number of steps needed to get the swarm

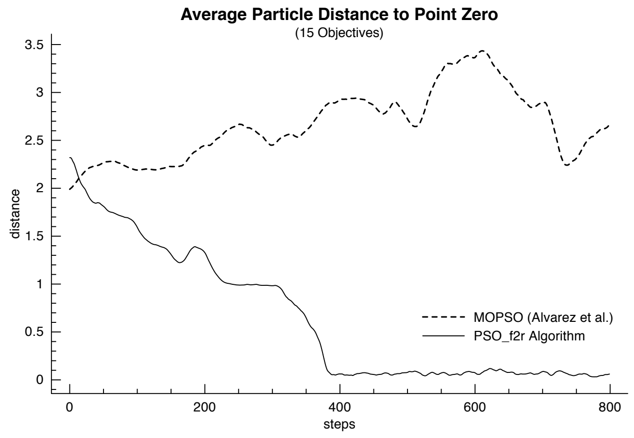

Figure 2: Average distances of the PSO$_{f2r}$ particles from the origin for the 15D-Pareto-Box case over 800 steps. For comparison, the distance values are also shown for another multi-objective PSO (of Alvarez et al. [1]), which is using Pareto dominance directly: in a 15-objective search-space, this algorithm performs like a random search.

within a 0.1 distance to the origin. Despite of the large fluctuation of these number of steps (given by the additionally provided standard deviation σ, and the maximum and minimum number found at 20 independent runs) and the interesting fact that for $n = 7$ the algorithm performance increases temporarily, from $n = 15$ on the search effort increases rapidly. The swarm spends more and more time in these relative extrema, and the number of these extrema is increasing exponentially with 2^n.

Compared to the study given in [14], the swarm needs an order of magnitude larger number of fitness evaluations than the FPD-GA (i.e. the Standard Genetic Algorithm transformed by the FPD to a multi-objective optimization algorithm) to approach the point 0, but solves the problem in cases where other EMOs (exemplified by the most popular NSGA-II) already fails. For FPD-GA, around 2000 fitness evaluations in the 20-dimensional case with a population of 10 individuals were reported, as compared to the values in table 1 with a population size of 7 particles. Given that, we can state the PSO$_{f2r}$ a higher search space exploration potential but a weaker exploitation. This is known for single-optimization PSOs for long, so we note the conceptional preservation of fundamental PSO properties in the multi-objective case when using the FPD paradigm to define the multi-objective PSO.

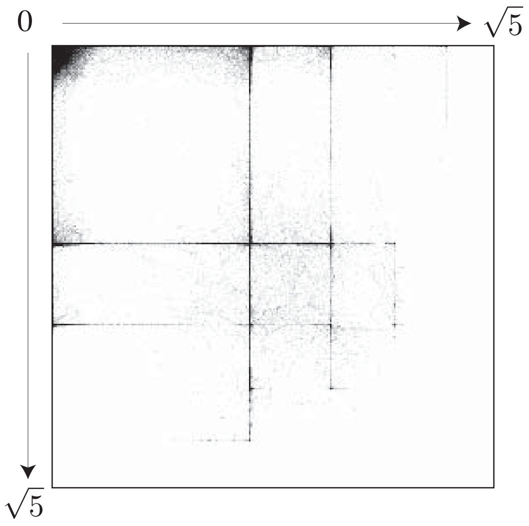

Figure 3: Search space exploration by a PSO_{f2r} applied to the 10D-Pareto-Box-Problem, 30 runs over 500 steps. The figure shows the particle position distribution from these 15000 samples in balanced coordinates.

5 Conclusions

Recently, the so-called Fuzzy-Pareto-Dominance (FPD) was proposed as a meta-heuristic to "convert" population-based single-objective optimization algorithms to multi-objective optimization algorithms. FPD is a generic ranking scheme, where ranking values are mapped to element vectors of a set. These ranking values are directly computed from the element vectors of the set and can be used to perform rank operations (e.g. selecting the "largest") with the vectors within the given set. In this paper, we introduced and explored the application of this concept to PSO, where a swarm of particles is maintained. The resulting PSO_{f2r} algorithm was studied on a fundamental optimization problem (so-called Pareto-Box-Problem) where a complete analysis can be provided. The PSO_{f2r} algorithm was shown to handle the case of a larger number of objectives, and shows similar properties like the (single-objective) PSO, as better exploration than exploitation capability, and local extrema entrapment.

Notable is the identification of a relative PARETO front as a pure consequence of the algorithm itself. The PSO_{f2r} algorithm can only leave a path leading to hyper-cube corners by adding small fluctuations to its particle velocities, controlled by so-called curl parameter (this fluctuation has been introduced for PSO by other authors as well). The PSO_{f2r} algorithm could be successfully validated on the Pareto-Box-Problem. However, the intricate influence of all parameter settings on the performance have to be much more studied, esp. to handle cases with more than 15 objectives better. An extension of the algorithm to escape from the identified relative extrema could become necessary. Then,

Table 1: Average number of steps needed to approach the origin within 10% for the first time, listed for several values of the number of objectives.

dimension	average	σ	min	max
2	42.1	19.9	22	111
5	159.4	123.2	50	473
7	133.2	63.2	52	273
10	291.0	124.9	92	592
15	600.9	283.7	276	1410
17	833.6	424.1	403	2154

the algorithm seems to be fit to be applied to real-world cases including multi-objective combinatorial optimization.

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] Thomas Bartz-Beielstein, Philipp Limbourg, Konstantinos E. Parsopoulos, Michael N. Vrahatis, Jörn Mehnen, and Karlheinz Schmitt. Particle Swarm Optimizers for Pareto Optimization with Enhanced Archiving Techniques. In *Proceedings of the 2003 Congress on Evolutionary Computation (CEC'2003)*, volume 3, pages 1780–1787, Canberra, Australia, December 2003. IEEE Press.

[3] U. Baumgartner, Ch. Magele, and W. Renhart. Pareto Optimality and Particle Swarm Optimization. *IEEE Transactions on Magnetics*, 40(2):1172–1175, March 2004.

[4] R. Brits, A.P. Engelbrecht, and F. van den Bergh. A Niching Particle Swarm Optimizer. In Lipo Wang, Kay Chen Tan, Takeshi Furuhashi, Jong-Hwan Kim, and Xin Yao, editors, *Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution and Learning (SEAL'02)*, volume 2, pages 692–696, Orchid Country Club, Singapore, November 2002. Nanyang Technical University.

[5] M. Clerc. The Swarm and the Queen: Towards a Deterministic and Adaptive Particle Swarm Optimization. In *Proc. of IEEE International Conference on Evolutionary Computation (ICEC'99), 1999*, 1999.

[6] 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.

[7] Carlos A. Coello Coello, Gregorio Toscano Pulido, and Maximino Salazar Lechuga. Handling Multiple Objectives With Particle Swarm Optimization. *IEEE Transactions on Evolutionary Computation*, 8(3):256–279, June 2004.

[8] Xiaohui Hu and Russell Eberhart. Multiobjective Optimization Using Dynamic Neighborhood Particle Swarm Optimization. In *Congress on Evolutionary Computation (CEC'2002)*, volume 2, pages 1677–1681, Piscataway, New Jersey, May 2002. IEEE Service Center.

[9] Evan J. Hughes. Multi-Objective Evolutionary Guidance for Swarms. In *Congress on Evolutionary Computation (CEC'2002)*, volume 2, pages 1127–1132, Piscataway, New Jersey, May 2002. IEEE Service Center.

[10] Evan J. Hughes. Swarm Guidance using a Multi-Objective Co-evolutionary On-Line Evolutionary Algorithm. In *2004 Congress on Evolutionary Computation (CEC'2004)*, volume 2, pages 2357–2363, Portland, Oregon, USA, June 2004. IEEE Service Center.

[11] Xiaohui Hui, Russell C. Eberhart, and Yuhui Shi. Particle Swarm with Extended Memory for Multiobjective Optimization. In *2003 IEEE Swarm Intelligence Symposium Proceedings*, pages 193–197, Indianapolis, Indiana, USA, April 2003. IEEE Service Center.

[12] J. Kennedy and R.C. Eberhart. Particle Swarm Optimization. In *IEEE International Conference on Neural Networks, Perth, Australia, IEEE Service Center, Piscataway, NJ, 1995*, 1995.

[13] James Kennedy and Russell C. Eberhart. *Swarm Intelligence*. Morgan Kaufmann Publishers, San Francisco, California, 2001.

[14] 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.

[15] 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.

[16] Xiaodong Li. A Non-dominated Sorting Particle Swarm Optimizer for Multiobjective Optimization. In Erick Cantú-Paz et al., editor, *Genetic and Evolutionary Computation—GECCO 2003. Proceedings, Part I*, pages 37–48. Springer. Lecture Notes in Computer Science Vol. 2723, July 2003.

[17] Xiaodong Li. Better Spread and Convergence: Particle Swarm Multiobjective Optimization Using the Maximin Fitness Function. In Kalyanmoy Deb et al., editor, *Genetic and Evolutionary Computation–GECCO 2004. Proceedings of the Genetic and Evolutionary Computation Conference. Part I*, pages 117–128, Seattle, Washington, USA, June 2004. Springer-Verlag, Lecture Notes in Computer Science Vol. 3102.

[18] Mahdi Mahfouf, Min-You Chen, and Derek Arturh Linkens. Adaptive Weighted Particle Swarm Optimisation for Multi-objective Optimal Design of Alloy Steels. In *Parallel Problem Solving from Nature - PPSN VIII*, pages 762–771, Birmingham, UK, September 2004. Springer-Verlag. Lecture Notes in Computer Science Vol. 3242.

[19] Sanaz Mostaghim and Jürgen Teich. Strategies for Finding Good Local Guides in Multi-objective Particle Swarm Optimization (MOPSO). In *2003 IEEE Swarm Intelligence Symposium Proceedings*, pages 26–33, Indianapolis, Indiana, USA, April 2003. IEEE Service Center.

[20] Sanaz Mostaghim and Jürgen Teich. Covering Pareto-optimal Fronts by Subswarms in Multi-objective Particle Swarm Optimization. In *2004 Congress on Evolutionary Computation (CEC'2004)*, volume 2, pages 1404–1411, Portland, Oregon, USA, June 2004. IEEE Service Center.

[21] K.E. Parsopoulos and M.N. Vrahatis. Particle Swarm Optimization Method in Multiobjective Problems. In *Proceedings of the 2002 ACM Symposium on Applied Computing (SAC'2002)*, pages 603–607, Madrid, Spain, 2002. ACM Press.

[22] Tapabrata Ray and K.M. Liew. A Swarm Metaphor for Multiobjective Design Optimization. *Engineering Optimization*, 34(2):141–153, March 2002.

[23] Y.H. Shi and R.C. Eberhart. A Modified Particle Swarm Optimizer. In *IEEE International Conference on Evolutionary Computation, Anchorage, Alaska, 1998*, 1998.

[24] 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.

[25] Gregorio Toscano Pulido and Carlos A. Coello Coello. Using Clustering Techniques to Improve the Performance of a Particle Swarm Optimizer. In Kalyanmoy Deb et al., editor, *Genetic and Evolutionary Computation–GECCO 2004. Proceedings of the Genetic and Evolutionary Computation Conference. Part I*, pages 225–237, Seattle, Washington, USA, June 2004. Springer-Verlag, Lecture Notes in Computer Science Vol. 3102.

[26] L.B. Zhang, C.G. Zhou, X.H. Liu, Z.Q. Ma, and Y.C. Liang. Solving Multi Objective Optimization Problems Using Particle Swarm Optimization. In *Proceedings of the 2003 Congress on Evolutionary Computation (CEC'2003)*, volume 4, pages 2400–2405, Canberra, Australia, December 2003. IEEE Press.