Multi-objective Fitness-proportional Attraction Approach with Weights

Patrick Laack
Institute for Intelligent Cooperative Systems
Otto-von-Guericke University Magdeburg, Germany
Email: patrick.laack@st.ovgu.de

Heiner Zille
Institute for Intelligent Cooperative Systems
Otto-von-Guericke University Magdeburg, Germany
Email: heiner.zille@ovgu.de

Sanaz Mostaghim
Institute for Intelligent Cooperative Systems
Otto-von-Guericke University Magdeburg, Germany
Email: sanaz.mostaghim@ovgu.de

***Abstract*—This paper proposes a new multi-objective optimization algorithm that is called Fitness-Proportional Attraction with Weights (F-PAW). In contrast to many other approaches, this work was inspired by physics rather than biology. It is based on concepts from several methods, including the attraction principle of gravity from the Gravitational Search Algorithm (GSA), the weight sum approach from Multi-Objective Evolutionary Algorithm based on Decomposition (MOEA/D) as well as particle swarm optimization methods. These and other algorithms that were providing inspiration are introduced during the text and their techniques are investigated for the use in F-PAW. The performance of F-PAW is compared to three well-known multi-objective algorithms through an experiment on 16 common test problems taken from the WFG and DTLZ benchmarks. The results indicate two conclusions. On the one side, the proposed approach with the weight sum obtains a good diversity. On the other side, the currently implemented local search is lacking reliability and speed.**

***Index Terms*—Multi-objective optimization, Gravitational Search Algorithm, Particle Swarm Optimization, Evolutionary Algorithms**

I. Introduction

In the area of optimization, many problems can hardly be solved analytically. Therefore, metaheuristic methods, often inspired by phenomena observable in nature, have been developed. Examples for biologically inspired optimization methods are Evolutionary Algorithms (EA) and Particle Swarm Optimizers (PSO). Based on these two basic ideas, a number of methods have been developed and refined to solve single- and multi-objective problems.

A multi-objective problem is usually defined as seen in the following Equation 1.

$$\begin{aligned} &\text{minimize } (f_1(\vec{x}), f_2(\vec{x}), ..., f_m(\vec{x})) \\ &\text{subject to } \vec{x} \in S \end{aligned} \tag{1}$$

The functions $f_1, \cdots, f_m$ represent the objective functions. They are mapping the decision space $S \subset \mathbb{R}^n$ of the problem into the objective space. Each objective function is a mapping $f_i : \mathbb{R}^n \to \mathbb{R}$.

Since in multi-objective problems these objective function are usually conflicting, a total ordering on the solution candidates can not be determined. Instead, the concept of Pareto-optimality is often used to describe the optimal set of solutions (Pareto-Set) of a multi-objective problem. In the same way, many algorithms use the principle of Pareto-dominance to rate the quality of solutions in multi-objective optimization.

Among the existing optimization algorithms, those inspired by biological principles are widely used and studied. However, other than the inspirations from biology, some ideas from physical systems have been the source of inspiration. One of these is the Gravitational Search Algorithm (GSA) [13]. Little research has been performed for multi-objective gravitation-inspired methods. For single-objective problems [13] reports a good performanceof GSA. The goal of this paper is to study a new way to extend the concept of GSA to multiple objectives and to compare the performance of the new algorithm with the existing appraoches.

The remainder of this paper is structured as follows: Section II describes related algorithms that form the basics of our proposed method and describes previous results. Section III presents the proposed Fitness-Proportional Attraction with Weights (F-PAW) algorithm and describes it in detail. The decisions that led to its design are explained and possible dis/-advantages are discussed as well. The evaluation of F-PAW is then commenced in section IV. At last, a conclusion is drawn and possible future work is lined out in Section V.

II. Related Work

The multi-objective optimization algorithm proposed in this work was inspired by concepts from several existing methods. In this section, we will describe this related work and its components that are needed in the remainder of this paper.

A. Particle-Swarm Optimization

The Particle Swarm Optimization (PSO) is inspired by the natural swarming behavior of birds and fish [8]. The particles (solutions) "move" in a continuous way using the decision space due to velocities assigned to them. In every iteration, there are two points to which they get attracted. Those are their personal best position and that of the global best position, which is also called the leader.

PSO was adapted to multiple objectives by different researchers. The primary obstacle in such an adaptation is to find a good way to determine the global best respectively a

978-1-5090-0623-6/16/$31.00

group of leaders, because under the consideration of Pareto-optimality, there will be likely a number of candidates among which one or several leaders must be chosen. One of these adaptations is the Speed-constrained Multi-objective Particle Swarm Optimization (SMPSO) by Nebro et al. [10]. As the name indicates, the velocities of the particles are constrained by a coefficient specific to that algorithm, in order to prevent erratic movements due to high speeds, but without having an arbitrary speed limit.

The SMPSO creates an archive of particles which is used as a set of potential leaders based on crowding distance. The selection of the leader for a specific agent is conducted through a 1-round binary tournament between two randomly selected members of the archive. The crowding distance is the sum of the "absolute normalized difference[s] in the function values of two adjacent solutions" [2]. Consequently, the solution with the lowest crowding distance will be considered as the worst one. The particles on the boundaries of the solution set usually receive infinite crowding distance values.

B. MOEA/D

An evolutionary method of interest for this work is the Multi-Objective Evolutionary Algorithm based on Decomposition (MOEA/D) [15]. It splits multi-objective problems into a number of single-objective optimization problems that are then solved simultaneously. The process of creating multiple single-objective problems is called decomposition and is usually done by an aggregation function. The aggregation function in MOEA/D combines all objective function values into one single objective function, and the different single-objective problems are created by using different parameters in the aggregation.

Popular aggregation functions are for example the weight sum and the Tchebycheff approach, which are used to rate the fitness of solutions with the help of weight vectors. These weights usually try to achieve diversity of solutions within the objective space by preferring different objectives functions to a certain extend during the aggregation. The MOEA/D algorithm assigns one solution to each of the aggregated objective functions, and therefore to each weight vector respectively.

C. Gravitation-based Algorithms

The GSA [13] was developed for single-objective optimization and is inspired by the law of gravity rather than biological concepts. Similar to particles in PSO methods, GSA algorithms treat the parameter values of a solution as a position inside the decision space. In every iteration these agents are evaluated and assigned a fitness with respect to their objective values. Afterwards, a gravity value is calculated for each agent based on their fitness and the overall progression of the algorithm. The higher the fitness, the stronger the gravity will be, while at the same time the gravity will become weaker during the progression of the algorithm. The movements of the agents are based on the gravitational force they perceive from the other agents. In the beginning, all agents exert an attractive force proportional to their gravity, while during the further progress of the algorithm only the K best agents are considered in the calculations of acceleration. The number of K best decreases with the progression of the algorithm.

There are some algorithms that have adapted GSA to multiple objectives. The Multi-Objective Gravitational Search Algorithm (MOGSA) presented by Hassanzadeh and Rouhani [6] adds an archive, whose K best members are then exerting gravity, but cannot move anymore. Only the active particles are still able to move and they only receive force from the archive. If one of them is no longer dominated by the archive, that particle will be added to the archive itself. They also use a uniform mutation operator, which randomizes one decision variable by "add[ing] a signed random term to it" [6].

Another adaption is the Non-dominated Sorting Gravitational Search Algorithm (NSGSA) [12], which combines Non-dominated Sorting Genetic Algorithm (NSGA) and GSA. When the archive used in this algorithm is full and another solution is about to be added, the crowding distance is used to determine whether the solution is added and which one is replaced. Three more papers were found [5], [11], [14], in which GSA was extended to multiple objectives with the help of an archive and different mutation operators. While the approaches from [5] and [11] target specific application problems, [14] was designed as a general-purpose algorithm similar to MOGSA and NSGA.

All of the mentioned algorithms are reported in their respective papers to obtain performances close to or better than that of previously published algorithms like NSGA-II. In [6] and [14] three multi-objective problems with two to three decision variables were used in the evaluation. 12 test problems were used in [12], but no information regarding the number of variables was provided. Assuming that the problems were used as defined in their original publications, those numbers would have been between two and 30.

III. Fitness-proportional Attraction with Weights

The proposed algorithm and the thoughts that led to its design will be described and discussed in this section. The algorithm will use a new approach that is called Fitness-Proportional Attraction with Weights and abbreviated as F-PAW.

A. Description of F-PAW

The basic idea of F-PAW is a combination of certain elements of the GSA, MOEA/D and PSO. As PSO and GSA, F-PAW is regarding a solution in the decision space of the problem as an *agent* with a certain position within the space. New solutions are created by altering the positions of the agents. In contrast to normal PSO, but similar to GSA algorithms, the equations that determine movements of the agents are based on simplified attraction equations derived from gravity. The novel concept of extending this GSA-based algorithm to multiple objectives in F-PAW is to use weight vectors for achieving diversity and deal with the trade-off between objectives.

The usage of those weights differs from that seen in MOEA/D in the way that the weight vectors of F-PAW are assigned to the agents (i.e. solution candidates) instead of assigning solution candidates to optimization problems (i.e. weight vectors) as in MOEA/D. During the initialization phase of F-PAW each agent is assigned a weight vector to give each objective function a relative importance. These weight vectors are not changed as the agents move and stay the same for the rest of the algorithms runtime. Thus, the aim is that the moving directions of the agents are influenced by the weights they possess. The creation of evenly distributed weight vectors is conducted in the same way as in the original MOEA/D work [15], with the goal of letting the agents in F-PAW try to explore the different areas of the objective space evenly.

Algorithm 1: F-PAW

Input: Real-valued optimization problem P with n variables and m objectives, Number of agents b, Archive size as
Output: Archive A

```
Initialize Archive A
Initialize Agent population B
foreach am ϵ A do Evaluate(am)
while Termination condition not fulfilled do
    foreach bm ϵ B do Evaluate(bm)
    Normalize function values
    UpdateArchive(as, A, B)
    LocalSearch (A, n)
    CalculateAcceleration(as, A, b, B, P)
    foreach bm ϵ B do CalculateVelocity(bm)
    foreach bm ϵ B do CalculatePosition(bm)
end
foreach bm ϵ B do Evaluate(bm)
UpdateArchive(as, A, B)
return A
```

F-PAW is represented as pseudo code in Algorithm 1. It uses two solution sets - an archive and an agent population. In the initialization step, both populations consist of random solutions, i.e. agents with random positions in the decisions space. The agents also receive a random starting velocity. The archive members are evaluated immediately. Each iteration of F-PAW then consists of the following four phases: (a) evaluate the agents, (b) archive update, (c) local search, (d) fitness-proportional attraction of agents. In the following we will explain each of those steps in further detail.

a) Agent Evaluation: Each iteration of the algorithm begins with the evaluation of the agents after which the normalization of the objective space is computed. Both archive and agents are considered for the normalization.

b) Archive Update: Then the archive is updated with the newly evaluated agents using a non-dominated sorting procedure. As a secondary measure in this process, crowding distance (CD) is used with the extension that the extremal solutions receive CD values of two times the value of the highest non-infinite value. This has to be done because the

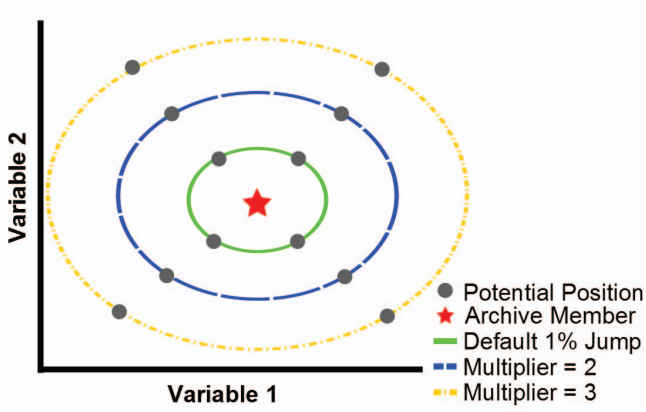

Figure 1. Potential positions of the 1st, 2nd, and 3rd clone of an archive member in two decision variables after the local search. As the ranges of the variables are not necessarily equal, they form ellipses.

F-PAW uses fitness-proportionality in the later computation of attraction, which would not be applicable with infinite values.

c) Local Search: The next phase is a local search step based on the solutions in the archive of the algorithm. The local search is visualized in Figure 1. The goal in this step is to let copies (clones) of selected solutions from the archive move with a fixed velocity in certain directions to look for neighboring good solutions.

For conducting this local search, a subset of k solutions is selected based on crowding distance from the archive population of the algorithm. For each of these solutions, a number of c clones are created. Each of these clone-agents is a solution with the same variable values as the respective archive solution.

Since an extensive local search in spaces of ≥ 10 dimensions would consume too many function evaluations, our local search mechanism will concentrate on only a subset of the total decision variables. A random selection is conducted among the decision variables of the problem to select μ variables (or all n variables if $n \leq \mu$).

All of the created clone-agents are receiving new velocities in the dimensions of the previously selected variables, while all other velocity dimensions are set to zero. When assigning the velocity values, the direction ($+$ or $-$) in the respective dimension is selected at random for each variable. By changing the absolute value of the set velocities, we can determine a *radius* of the local search, i.e. amount of change that will occour during the local search in each variable. This radius is determined by 2 factors. (1) We define a step size parameter S_i of 1% of the domain of variable i as a basic search radius. (2) We count the number of iterations in which a single archive solution has been used in the local search. The assigned velocities in the selected dimensions are the product of the respective step size S_i and the number of times a solution has participated in a local search. By increasing the search radius in consecutive local searches around the same agent, bigger search steps are possible when one of the k best archive members remains among the k best for several

iterations, which might indicate a local optimum and a bigger search step is necessary to leave it.

To account for possible concentrated areas of Pareto-optimal solutions towards the end of the search process, the parameter S_i is subject to change throughout the runtime of the algorithm. In addition to the above mentioned setting, in our implementation it will be set to 1% of the archives solutions range for that variable, if less than 25% of the total function evaluations remain.

Finally, to conduct the local search, the clones are moved according to their velocities and evaluated. A normal archive update as mentioned above is conducted with the resulting population of clones.

d) Fitness-Proportional Attraction: Initially, there will be a short period of time when there is no attraction at all, and the starting velocities of the agents make up all their movements. This short period is intended to help building the archive by providing more starting solutions. After this initial phase, F-PAW uses a fitness-proportional attraction mechanism based on GSA. Each agent is attracted to one leader solution from the archive. Based on this attraction, its acceleration is calculated and the velocities and positions are updated accordingly.

In every iteration, each agent gets his leader assigned from the archive based on a roulette wheel selection mechanism with stochastic universal sampling (SUS). SUS was published by Baker [1] and serves the purpose of reducing unfair bias towards the proportionally more fit archive members. Furthermore, it increases the computational efficiency by reducing the number of generated random values to one.

The weight vector of an agent, the vector of its leaders objective values, and the Euclidean distance between the two particles are then multiplied with each other. Summing up the values of the product vector yields the acceleration of the agent as can be seen in Equation 2.

$$a_i = \sum_{o=1}^{m} W_i(o) * F_j(o) * D_{ij} \tag{2}$$

Where a_i is the acceleration of agent i for this iteration, m is the number of objectives, $W_i(o)$ is the weight of the o-th objective for agent i, $F_j(o)$ is the fitness respectively the objective value of archive member j leading agent i in the o-th objective, and D_{ij} is the Euclidean distance between i and j in the decision space.

After calculating their accelerations, the velocities and positions of the agents are updated accordingly through vector addition. The previous velocities are multiplied with an inertia factor (e.g. 0.4) and the modifying acceleration has a random multiplier between zero and one. If the new position of an agent would be outside of the decision space, it is set onto the boundary in the dimension of that respective decision variable and its velocity is inverted and divided by ten.

Figure 2 summarizes these mechanisms of F-PAW. The main loop then either begins anew or ends when the termination condition of the algorithm is met.

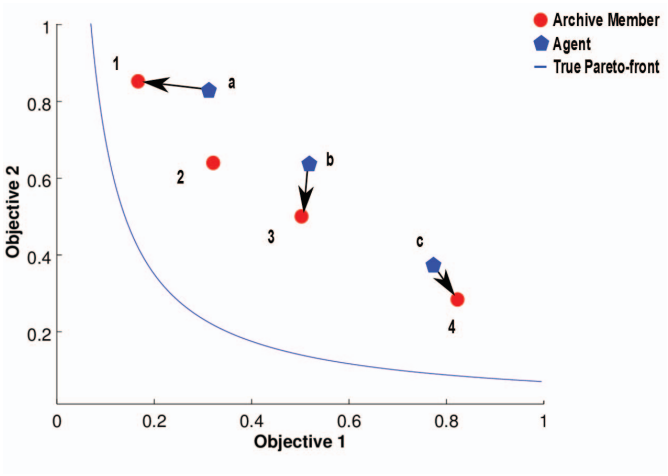

Figure 2. Schematic visualization of F-PAW showing four archive members followed by three agents. The archive members 1 and 4 would always be used in the local search.

B. Conception

As pointed out in the previous section, the algorithm should be able to deal with different properties of Pareto-fronts. Based on the assumption that good solutions in the objective space can be found in the neighbourhood of other good solutions in the decision space, a local search was deemed suitable and implemented. Whilst this might suffice for linear, convex and concave problems, it is not promising regarding disconnected and multi-modal problems. To counter this expectation, a dynamic multiplier was added, which increases the range of the local search to escape local optima.

To limit the computational effort, only the k best archive members are used in the local search step. This is partially inspired by the original GSA [13], but with the difference that k does not change over time. The local search is thus a second mechanism of exploration in F-PAW. It should always include the two outermost members of the archive to spread it further. The limitation to only the best k agents is not only motivated by performance, but also to reduce redundancy. Conducting a local search around two solutions that are close to each other, will with certain likelihood lead to two new solutions that are again similar to each other.

While the local search extends and improves the boundaries of the archive, the remainder of the agents are unaffected or might get dominated. The aim of the (non-archive) agents is to fill this gap. Their attraction to the archive member is similar to that of MOGSA from Hassanzadeh and Rouhani [6]. But rather than gravity, the normalized objective values are used directly to calculate the acceleration.

The attraction function used in our approach differs from that of the original GSA and also by the physical inspiration. Instead of dividing by the (square of the) distance, the fitness is multiplied by it to have high speeds for distant agents and low velocities for close ones. The relation is inverted, because

Table I
CONFIGURATIONS AND PROPERTIES OF THE USED TEST PROBLEMS. MIXED STANDS FOR A PARETO-FRONT THAT IS PARTIALLY CONCAVE AND PARTIALLY CONVEX.

Problem	n	m	Known Properties
WFG1	6	2	Mixed, Biased
WFG2	6	2	Convex, Disconnected, Multi-modal, Non-separable
WFG3	6	2	Linear, Degenerate, Non-separable
WFG4	6	2	Concave, Multi-modal
WFG5	6	2	Concave, Deceptive
WFG6	6	2	Concave, Non-separable
WFG7	6	2	Concave, Biased
WFG8	6	2	Concave, Biased, Non-separable
WFG9	6	2	Concave, Biased, Multi-modal, Deceptive, Non-separable
DTLZ1	7	3	Linear, Multi-modal
DTLZ2	12	3	Concave
DTLZ3	12	3	Concave, Multi-modal
DTLZ4	12	3	Concave, Biased
DTLZ5	12	3	Concave, Degenerate
DTLZ6	12	3	Concave, Degenerate, Multi-modal
DTLZ7	22	3	Mixed, Disconnected, Multi-modal

a higher acceleration due to smaller distance could be expected to make the agents 'jump' over their leaders.

A similarity to SMPSO [10] is the leader selection that is used. A crowding-distance-proportional roulette wheel with SUS determines which agent follows which archive member. In this way, we aim to improve the diversity of our obtained solution sets.

Another feature of F-PAW which is also used for diversity reasons is that each agent is assigned weights for the objectives during its initialization. This is inspired by the "weight sum approach" from MOEA/D. As described above, the weights are distributed evenly throughout the objective space. The goal of using the weight sum approach is also to obtain a better diversity among the agents, as they should spread according to their weights. This should consequently lead to a better diversity in the archive as well.

IV. EVALUATION

The investigation of F-PAW's performance is commenced in this section. We explain how the evaluation is structured, what its results are, and possible reasons for the results are given in the analysis.

A. Setup of the Experiment

F-PAW was implemented and integrated into the jMetal framework [4], [9]. Most test problems from the groups of Deb, Thiele, Laumanns, and Zitzler (DTLZ) [3] and Walking Fish Group (WFG) [7] are used. Problems that are not included in jMetal and therefore not used in this evaluation are DTLZ8 as well as 9. This leads to a total number of 16 test problems. The reference fronts from the jMetal website [9] are precomputed representations of the true Pareto-fronts and used for the comparison of the results delivered by the algorithms.

All test problems are used in their default configurations included in jMetal. The number of variables and objectives can be seen in Table I. It also provides an overview of their properties, but these lists are not to be considered complete. The information was obtained from their corresponding papers.

F-PAW is compared to NSGA-II, MOEA/D, and SMPSOas they (except NSGA-II) have similarities to F-PAW. Table Different indicators for the evaluation of multi-objective optimization algorithms are available. The first measure we use is the hypervolume [17], which measures the convergence and diversity of a solution set. For the hypervolume indicator, a larger value indicates better performance. The second indicator we use here is the generalized Δ-measure proposed by Deb et al. [2] and later extended by Zhou et al. [16] to be applicable to more than two objectives. It aims to measure the diversity of a solution set. For the Δ-measure, smaller values indicate better performance of an algorithm.

B. Parameter Settings

The evaluated algorithms have several settings that can affect their performances. In order to make the experiment reproducible, at least as much as the included randomness allows, those settings are described here. For each of the tested algorithms and benchmark problems, we perform 100 independent runs. The maximum number of evaluations per run of any algorithm is set to 25000.

The size of the population for MOEA/D and NSGA-II is set to 120. F-PAW uses a population size of 30 and SMPSO of 100. The archive size however is 120 for SMPSO and F-PAW too. MOEA/D and NSGA-II have no archives. If an algorithm uses an archive, its population size was left at the default value. In the case of F-PAW a well working value is taken, because a difference in those values does not interfere with comparability.

The NSGA-II, MOEA/D and SMPSO algorithms use mutation operators. The mutation probability is set to $1/n$ (with n being the number of decision variables in the problem) and the mutation distribution index is set to 20.0. All of them use a polynomial mutation operator. NSGA-II has its crossover probability set to 0.9 and a crossover distribution index of 20.0. It uses an SBX crossover operator. MOEA/D has its differential evolution crossover operator configured with a control value of 1.0 and a scaling factor for mutation of 0.5.

MOEA/D requires files that provide weight vectors depending on the population size and the number of objectives. They were generated beforehand and are evenly spread throughout the respective objective spaces. The MOEA/D implementation uses the Tchebycheff aggregation approach. MOEA/D has three more settings. Its neighbourhood size is set to 20, the probability that parent solutions are selected from neighborhood is set to 0.9, and the maximal number of solutions replaced by each child solution is 2.

There are four more parameters to set in F-PAW. The inertia factor reduces the velocities of the agents in each iteration. This keeps the speeds down and prevents erratic movements in the agents. However, the same feature might also slow down the convergence of the algorithm as a whole. In our experiments, we set this value to 0.4. Another part that can

Table II
HYPER-VOLUME. MEAN AND STANDARD ERROR. DARK GRAY INDICATES THE BEST VALUE. LIGHT GRAY INDICATES THE SECOND BEST.

	NSGA-II	MOEA/D	SMPSO	F-PAW
WFG1	$5.16e-01$	$3.24e-01$	$1.17e-01$	$2.54e-01$
	$8.3e-03$	$8.2e-03$	$6.1e-04$	$5.1e-03$
WFG2	$5.62e-01$	$5.62e-01$	$5.61e-01$	$5.64e-01$
	$1.4e-04$	$3.5e-05$	$7.4e-05$	$7.9e-05$
WFG3	$4.41e-01$	$4.42e-01$	$4.41e-01$	$4.42e-01$
	$2.9e-05$	$5.8e-06$	$1.7e-05$	$6.1e-05$
WFG4	$2.18e-01$	$2.12e-01$	$2.03e-01$	$2.14e-01$
	$2.9e-05$	$1.8e-04$	$1.9e-04$	$2.0e-04$
WFG5	$1.96e-01$	$1.96e-01$	$1.97e-01$	$2.12e-01$
	$7.2e-05$	$8.2e-05$	$4.2e-06$	$5.5e-04$
WFG6	$2.01e-01$	$2.10e-01$	$2.10e-01$	$2.08e-01$
	$9.5e-04$	$1.1e-05$	$2.9e-05$	$2.7e-04$
WFG7	$2.10e-01$	$2.10e-01$	$2.10e-01$	$2.11e-01$
	$2.1e-05$	$8.5e-06$	$2.1e-05$	$3.8e-05$
WFG8	$1.53e-01$	$1.62e-01$	$1.48e-01$	$1.41e-01$
	$1.3e-03$	$2.1e-03$	$1.8e-04$	$1.3e-03$
WFG9	$2.38e-01$	$2.36e-01$	$2.36e-01$	$2.36e-01$
	$1.7e-04$	$6.7e-05$	$4.4e-05$	$3.2e-04$
DTLZ1	$5.66e-01$	$6.41e-01$	$7.41e-01$	$5.10e-02$
	$2.6e-02$	$2.3e-02$	$4.2e-03$	$1.4e-02$
DTLZ2	$3.82e-01$	$3.84e-01$	$3.59e-01$	$3.96e-01$
	$5.3e-04$	$2.3e-04$	$6.4e-04$	$3.9e-04$
DTLZ3	$0.00e+00$	$1.81e-01$	$3.18e-01$	$0.00e+00$
	$0.0e+00$	$1.7e-02$	$7.3e-03$	$0.0e+00$
DTLZ4	$3.83e-01$	$3.56e-01$	$3.61e-01$	$3.92e-01$
	$4.8e-04$	$3.7e-03$	$1.8e-03$	$4.8e-04$
DTLZ5	$9.34e-02$	$9.01e-02$	$9.42e-02$	$9.44e-02$
	$1.7e-05$	$9.1e-06$	$7.4e-06$	$6.5e-06$
DTLZ6	$0.00e+00$	$9.13e-02$	$9.53e-02$	$6.74e-02$
	$0.0e+00$	$1.8e-06$	$4.8e-06$	$3.6e-03$
DTLZ7	$2.83e-01$	$1.57e-01$	$2.81e-01$	$2.12e-01$
	$4.0e-04$	$3.1e-03$	$5.1e-04$	$3.6e-03$

Table III
GENERALIZED Δ-MEASURE. MEAN AND STANDARD ERROR. DARK GRAY INDICATES THE BEST VALUE. LIGHT GRAY INDICATES THE SECOND BEST.

	NSGA-II	MOEA/D	SMPSO	F-PAW
WFG1	$6.59e-01$	$8.45e-01$	$9.60e-01$	$9.76e-01$
	$1.1e-02$	$1.0e-02$	$4.4e-03$	$1.0e-02$
WFG2	$4.83e-01$	$7.99e-01$	$3.10e-01$	$1.54e-01$
	$1.1e-02$	$5.0e-03$	$5.3e-03$	$2.8e-03$
WFG3	$6.03e-01$	$5.67e-01$	$3.90e-01$	$3.70e-01$
	$2.9e-03$	$2.2e-04$	$5.7e-04$	$6.4e-04$
WFG4	$3.46e-01$	$5.05e-01$	$5.31e-01$	$2.10e-01$
	$4.0e-03$	$4.1e-03$	$6.6e-03$	$4.7e-03$
WFG5	$3.93e-01$	$4.72e-01$	$1.38e-01$	$4.11e-01$
	$3.8e-03$	$1.5e-03$	$1.3e-03$	$1.5e-02$
WFG6	$3.66e-01$	$4.22e-01$	$1.44e-01$	$1.23e-01$
	$3.7e-03$	$8.4e-04$	$1.5e-03$	$2.0e-03$
WFG7	$3.72e-01$	$4.19e-01$	$1.43e-01$	$1.25e-01$
	$3.6e-03$	$7.5e-04$	$1.4e-03$	$1.3e-03$
WFG8	$5.46e-01$	$5.79e-01$	$6.49e-01$	$5.45e-01$
	$4.9e-03$	$3.6e-03$	$1.0e-02$	$1.2e-02$
WFG9	$3.53e-01$	$4.90e-01$	$1.77e-01$	$1.33e-01$
	$3.8e-03$	$2.6e-03$	$1.8e-03$	$2.6e-03$
DTLZ1	$7.22e-01$	$7.12e-01$	$3.87e-01$	$7.64e-01$
	$3.2e-02$	$4.4e-03$	$9.5e-03$	$1.8e-02$
DTLZ2	$5.08e-01$	$8.03e-01$	$3.91e-01$	$3.81e-01$
	$3.6e-03$	$2.2e-03$	$3.2e-03$	$3.0e-03$
DTLZ3	$9.93e-01$	$7.98e-01$	$6.68e-01$	$1.09e+00$
	$2.4e-02$	$7.2e-03$	$4.3e-02$	$1.2e-02$
DTLZ4	$4.93e-01$	$9.80e-01$	$5.29e-01$	$3.99e-01$
	$4.6e-03$	$1.2e-02$	$1.4e-02$	$4.8e-03$
DTLZ5	$4.31e-01$	$1.23e+00$	$1.21e-01$	$1.24e-01$
	$5.4e-03$	$8.1e-03$	$1.5e-03$	$1.7e-03$
DTLZ6	$6.39e-01$	$1.31e+00$	$1.08e-01$	$6.32e-01$
	$4.1e-03$	$7.4e-03$	$1.9e-03$	$2.3e-02$
DTLZ7	$4.89e-01$	$9.04e-01$	$4.53e-01$	$4.95e-01$
	$3.8e-03$	$6.8e-03$	$4.5e-03$	$9.4e-03$

be configured is the local search. The number of the k best archive members to be cloned and the amount of clones created per chosen agent best that can be modified. In this experiment they are set to $k = 5$ and $c = 1$ respectively.

Furthermore, the threshold for the search step size can be altered. As mentioned earlier, it changes the maximum step size to one percent of the ranges of the archive, when the algorithm has used a portion of its available function evaluations. For this experiment, that threshold has been set to 75%.

As mentioned earlier, there is an initial phase during which the agents are not attracted by the archive. In our experiments, we set this parameter to the 3% percent of the algorithm's total function evaluations. During this period, the starting velocities make up all their movements.

C. Results and Analysis

Tables II and III show the means and standard errors for chosen evaluation measures hypervolume and Δ-measure. A dark gray field indicates the best value among the five algorithms for that test problem and a light gray fields shows the second best performance.

F-PAW obtains the best performance in terms of hypervolume (Table II) for the test problems WFG2, 5, and 7 as well as for DTLZ2, 4 and 5. For WFG3 and WFG4 it performs second best. All four algorithms' results differ only slightly for these problems with differences mostly being less than 5%. For DTLZ3 the calculated hypervolume values of F-FAW and NSGA-II were zero due to worse performance.

Taking a look at the Δ-measure indicator, we can observe that F-PAW performs better than in terms of hypervolume, as seen in Table III. Here F-PAW is the best performing algorithm for the WFG2, 3, 4, 6, 7, 8, and 9 as well as for the DTLZ2 and 4 test problems. It also obtains the second best Δ-measure values for DTLZ5 and 6. The strongest competitor with respect to the Δ-measure indicator is the SMPSO algorithms with 6 best and 6 second best scores.

The hypervolume indicates that F-PAW's convergence is on par with the compared algorithms in most test problems. The convergence is unreliable in some problems like DTLZ3, which indicates that there is a difficulty for the local search. Testing has shown that the convergence can be improved for most problems by increasing the number of available evaluations. This might point towards a slower convergence rate of F-PAW compared to the other algorithms.

The stronger side of F-PAW is its performance in terms of diversity. We can observe this by looking at the Δ-measure indicator in Table III. The distribution of weights is likely the biggest contributing component for the diversity, because the agents are bound to spread evenly, as long as the values for

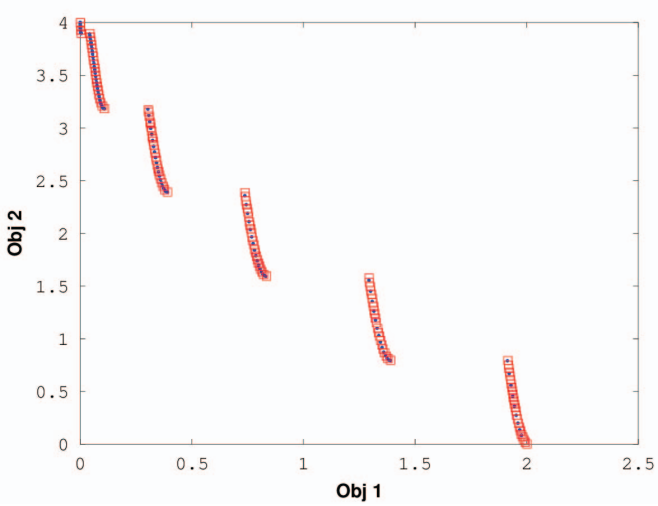

Figure 3. One run of F-PAW on WFG2. The blue points show a subset of the Pareto-front and the red squares are the final archive members of F-PAW in that run.

the crowding distances of the archive members do not have too big differences and there are enough archive members to begin with.

For some problems F-PAW performs best or second best in both indicators. Those are WFG2, 3, 4 and 7 as well as DTLZ2, 4, and 5. They will henceforth be termed as *well solved* for the remainder of this evaluation. Looking at the properties from Table I, these problems have no single common property that is present in all of them other than having multiple objectives. And as some of them have two and others have three objectives, this commonality is also limited.

For instance, WFG2 has a convex Pareto-front, WFG3 a linear one, and the remaining well solved problems have concave Pareto-fronts (e.g. DTLZ2). The only shape missing among the well solved problems is a mixed front like those of WFG1 or DTLZ7. On the former F-PAW can reach a good but partial convergence and a bad diversity of the Pareto-front or a local front with acceptable convergence and diversity. It can therefore be concluded that the shape of a problem's front is either irrelevant to F-PAW or that it may have difficulties with mixed fronts.

Regarding concave Pareto-fronts, some research showed that using a weight sum approach can have disadvantages when dealing with concave fronts. This leads to the expectation that, based on the attraction formula of F-PAW, there might be difficulties in our experiments for these cases too. However, in the results we could not observe this, as there are five concave problems among the well solved ones.

If we are looking at the two problems with disconnected fronts (WFG2 and DTLZ7), only the first one is well solved. The obtained solution set of an exemplary run by F-PAW is shown in Figure 3. For DTLZ7 the results are still similar to those of the other algorithms. But looking at visualizations shows that F-PAW only occasionally does reach all four parts of the Pareto-front, while it always finds at least one of them.

This concurs with the expectation that F-PAW is not performing well on problems with optimal regions that are too far apart. It is also a logical consequence of the fact, that the local search becomes increasingly ineffective with bigger search steps, because the number of evaluated solutions in the neighbourhood of an archive member stays the same even when the local search space becomes an ever bigger hypersphere around the archive member.

The two multi-modal problems that are well solved are WFG2 and 4. All other multi-modal problems are not closely approximated by F-PAW. Though it should be noted that the well solved WFG5 should be multi-modal as well, because it is deceptive. Howver, its degree of multi-modality might be negligible. Another observation which also shows that F-PAW is struggling with multi-modality are the differences in the results between the problems DTLZ2 and DTLZ3. That is, because DTLZ3 is a multi-modal variant of the DTLZ2. F-PAW reaches points that are near optimal for two objectives, but it fails to do so for all three objective functions. The results of DTLZ1 are better, though F-PAW often gets stuck in a local front.

WFG2, 3, and 6 are non-separable and F-PAW obtains good results when optimizing them. This property does not appear to influence the results negatively as can be expected. The same can be said for the biased WFG7 and DTLZ4, for which F-PAW's results (see Figure 4) do not look very different from those of the base problem DTLZ2. However, F-PAW does not perform in the same way for the WFG8 problem, even though WFG8 only differs from the well solved WFG4 by being non-separable and biased.

The well solved and degenerate WFG3 and DTLZ5 problems indicate that the case of degenerated Pareto-fronts is without greater influence to F-PAWs performance as well. Though DTLZ5 might only provide limited information here, as its Pareto-front is unknown for more than three objectives.

The last feature that remains is deceptiveness. WFG5 is a deceptive problem and F-PAW obtains good convergence and mediocre to good diversity in it. For WFG9, which is deceptive as well, it reaches the best diversity and a convergence that is close to the best performance of NSGA-II for this problem. It can therefore be concluded that F-PAW can solve deceptive problems without greater difficulties.

V. Conclusions and Future Work

In this work we proposed a new optimiaztion algorithm called Fitness-Proportional Attraction with Weights. This approach is inspired by previous gravitational search algorithms, while at the same time tries to combine several promising components from Particle Swarm Optimization and Evolutionary Algorithms. F-PAW should not be seen as a purely gravitation-based algorithm because of the similarities to many other approaches. It can be regarded as a hybrid between concepts from MOGSA, SMPSO, MOEA/D, and a novel approach.

The conducted experiments show that our proposed implementation of the F-PAW algorithm performs well in at least one problem for each identified feature and can therefore be

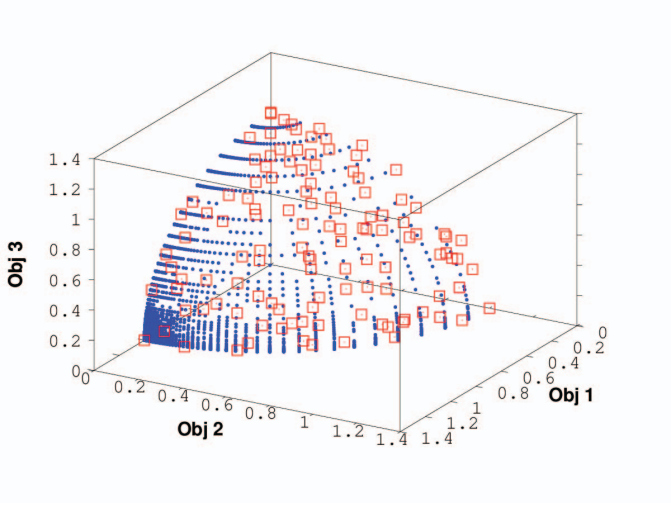

Figure 4. One run of F-PAW on DTLZ4. The blue points show a subset of the Pareto-front and the red squares are the final archive members of F-PAW in that run.

considered as a general-purpose algorithm. The good diversity in the results might be interesting for some applications. Especially when convergence is not that important or for a possible post-optimization of the already well converged results obtained with another method.

Future research on F-PAW could be an extension to solve many-objective problems with four or more objective functions. Additionally, F-PAW should be optimized in the aspects of computational runtime and convergence rate in order to obtain a higher efficiency and effectiveness. A specific point that might lead to gains in those aspects could be a modified leader selection for the acceleration calculation of the population agents.

References

[1] J. E. Baker. Reducing Bias and Inefficiency in the Selection Algorithm. In *Proceedings of the Second International Conference on Genetic Algorithms on Genetic Algorithms and Their Application*, pages 14–21, Hillsdale, NJ, USA, 1987. L. Erlbaum Associates Inc.

[2] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *Evolutionary Computation, IEEE Transactions on*, 6(2):182–197, 2002.

[3] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler. Scalable multi-objective optimization test problems. In *Evolutionary Computation, 2002. CEC '02. Proceedings of the 2002 Congress on*, volume 1, pages 825–830, 2002.

[4] J. J. Durillo and A. J. Nebro. jMetal: A Java framework for multi-objective optimization. *Advances in Engineering Software*, 42(10):760–771, 2011.

[5] D. L. González-Álvarez, M. A. Vega-Rodríguez, J. A. Gómez-Pulido, and J. M. Sánchez-Pérez. Applying a Multiobjective Gravitational Search Algorithm (MO-GSA) to Discover Motifs. In J. Cabestany, I. Rojas, and G. Joya, editors, *Advances in Computational Intelligence*, volume 6692 of *Lecture Notes in Computer Science*, pages 372–379. Springer Berlin Heidelberg, 2011.

[6] H. R. Hassanzadeh and M. Rouhani. A Multi-objective Gravitational Search Algorithm. *Computational Intelligence, Communication Systems and Networks, International Conference on*, 0:7–12, 2010.

[7] S. Huband, P. Hingston, L. Barone, and L. While. A review of multiobjective test problems and a scalable test problem toolkit. *Evolutionary Computation, IEEE Transactions on*, 10(5):477–506, 2006.

[8] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Neural Networks, 1995. Proceedings., IEEE International Conference on*, volume 4, pages 1942–1948 vol.4, 1995.

[9] A. J. Nebro and J. J. Durillo. jMetal Web site. http://jmetal.sourceforge.net/index.html.

[10] A. J. Nebro, J. J. Durillo, J. García-Nieto, C. A. Coello Coello, F. Luna, and E. Alba. SMPSO: A New PSO-based Metaheuristic for Multi-objective Optimization. In *2009 IEEE Symposium on Computational Intelligence in Multicriteria Decision-Making (MCDM 2009)*, pages 66–73. IEEE Press, 2009.

[11] T. Niknam, M. R. Narimani, R. Azizipanah-Abarghooee, and B. Bahmani-Firouzi. Multiobjective Optimal Reactive Power Dispatch and Voltage Control: A New Opposition-Based Self-Adaptive Modified Gravitational Search Algorithm. *Systems Journal, IEEE*, 7(4):742–753, 2013.

[12] H. Nobahari, M. Nikusokhan, and P. Siarry. A Multi-Objective Gravitational Search Algorithm Based on Non-Dominated Sorting. *International Journal of Swarm Intelligence Research (IJSIR)*, 3(3):32–49, 2012.

[13] E. Rashedi, H. Nezamabadi-pour, and S. Saryazdi. GSA: A Gravitational Search Algorithm. *Information Sciences*, 179(13):2232–2248, 2009.

[14] S. Tabatabaei. A new gravitational search optimization algorithm to solve single and multiobjective optimization problems. *Journal of Intelligent and Fuzzy Systems*, 26(2):993–1006, 2014.

[15] Q. Zhang and H. Li. MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition. *Evolutionary Computation, IEEE Transactions on*, 11(6):712–731, 2007.

[16] A. Zhou, Y. Jin, Q. Zhang, B. Sendhoff, and E. Tsang. Combining Model-based and Genetics-based Offspring Generation for Multi-objective Optimization Using a Convergence Criterion. In *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*, pages 892–899, 2006.

[17] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. *Evolutionary Computation, IEEE Transactions on*, 3(4):257–271, 1999.