DISAGREEMENTS – A NEW SOCIAL CONCEPT IN SWARM INTELLIGENCE AND EVOLUTIONARY COMPUTATION

Teză destinată obţinerii
titlului ştiinţific de doctor inginer
la
Universitatea "Politehnica" din Timişoara
în domeniul ŞTIINŢA CALCULATOARELOR
de către

Ing. Andrei Lihu

Conducător ştiinţific:	prof. univ. dr. ing. Ştefan Holban
Referenţi ştiinţifici:	prof. univ. dr. ing. Dumitru Dan Burdescu
	prof. univ. dr. Alexandru Cicortaş
	prof. univ. dr. ing. Marius Crişan

Ziua susţinerii tezei: 10.02.2012

România, 300159 Timişoara, Bd. Republicii 9,
tel. 0256 403823, fax. 0256 403221
e-mail: editura@edipol.upt.ro

Foreword

Optimization is a process of searching for desirable configurations, attributes or parameters in a system. Unfortunately, for most of the real-world problems that need to be optimized, this search procedure is requiring heavy computational resources. Therefore, in order to tackle this complexity, usually some heuristics are employed.

Swarm intelligence algorithms like *particle swarm optimization* (PSO) and evolutionary techniques like *genetic algorithms* (GAs) are very popular choices as optimization heuristics and they are both population-based models. Particle swarm optimization models "the social mind" metaphor: the solutions are found by flying particles that are influenced by their personal and their group's experience. Since the inception of the original algorithm in 1995, the scientific literature regarding swarm intelligence grew massively. Although some hybrids between PSO and other optimization techniques were invented and an extensive analysis of parameters was thoroughly done, the research in this field did not develop and incorporate other basic social characteristics into the original algorithm.

This PhD thesis aims to fill the lack of other social behaviors in PSO. Based on real-world observations, I developed some models that resemble the way people disagree inside a group and implemented it as a new social feature in PSO. The idea was validated by good experimental results and then I decided to apply it as a special mutation operator in real-valued genetic algorithms. The results were better than expected and prompted for an extensive and thorough investigation. The thesis is the result of this effort.

Timişoara, august 2011 Lihu Andrei

This work was partially supported by the strategic grant POSDRU 6/1.5/S/13, Project ID6998 (2008), co-financed by the European Social Fund Investing in People, within the Sectorial Operational Programme Human Resources Development 2007-2013.

Special thanks go to this thesis' scientific supervisor, Prof. Dr. Eng. Ştefan Holban, for promptness, patient guidance, good advice and encouragement provided throughout my time as a student.

Completing this work would have been a lot more difficult without the moral support from Oana, my fiancée. I also want to thank her for the patience and the help in proof reading countless pages of scientific and technical information.

Lihu, Andrei

Disagreements - A New Social Concept in Swarm Intelligence and Evolutionary Computation

Teze de doctorat ale UPT, Seria 10, Nr. 39, Editura Politehnica, 2012, 98 pagini, 35 figuri, 24 tabele.

ISSN: 1842-7707

ISBN: 978-606-554-439-0

Cuvinte cheie: global optimization, particle swarm optimization, genetic algorithms, disagreements, swarm intelligence, evolutionary computation.

Rezumat,
Based on observations from nature and promising experimental data, this work opens a new direction in the field of evolutionary computation: the usage of disagreements to derive superior algorithms with enhanced exploitation and exploration capabilities for global optimization problems. Initially, good results are obtained with the new 6σ-PSOD operator which imitates the natural distribution of the disagreements in nature with a low computational cost. The disagreements are applied only to the social component from PSO. The partial disagreements increase the exploitation, while the extreme disagreements enhance the exploration. Disagreements can also be used to mitigate stagnation in swarm with the new RS-PSOD operator. Further good results with the concept of disagreements on real-valued genetic algorithms open the door for experimenting with this new idea in the larger field of the evolutionary computation. This thesis is a proof of concept for disagreements in swarm intelligence, through PSO, and in the larger category of the evolutionary computation, through real-valued genetic algorithms. Because its contributions are at the core of the algorithms in study, the potential impact can positively affect all areas in which evolutionary computation is used.

Contents

List of Tables

List of Figures

1. INTRODUCTION

1.1. Motivation

Optimization processes happen everywhere in the universe, therefore both in nature and in human activities. We always search for and exploit the shortest paths that lead to a satisfactory solution. We live in a world in which transportation timetables, telecommunications bandwidth allocations, road networks and product manufacturing, to name just few examples, are usually designed to serve most needs quicker and in a more and more cost-effective manner. Since the needs of our society and the society itself continues to grow, it is clear why we need better optimization algorithms that can solve increasingly complex problems.

In time, research proved that heuristic methods, without a clear winner (No Free Lunch Theorem in [98, 99]), can be used to solve hard problems. A large category of optimization techniques is using a populational model that is evolving or moving in the solutions' search space using well established rules. Evolutionary and swarm intelligence algorithms are good examples that fit in this segment and prove successful for a wide range of domains and practical applications, like cancer therapy planning, economic and financial forecasting, industrial product design, data analysis, automated scheduling and so on. There is no doubt that even a small improvement in the area of optimization techniques will produce a big impact in so many fields of human activity. This is the reason why there is a high interest and a large number of researchers and publications on this subject.

Most of the time for these problems the domain information is sparse, and these methods are not guaranteed to find the global optimum, especially for NP-complete problems, but they can provide good enough approximate solutions when all other classical approaches fail. One of the big problems is that it happens quite often that many of these optimizers are getting trapped into insatisfactory local optima. This is a tedious task to solve, because it is generally hard to tell whether the best solution found so-far is the global optimum; one of the classic solutions to this is to increase diversity among the population. Other times, optimizers fail to find a solution or cannot find it in a reasonable amount of time, because they lack convergence. These two issues are mostly addressed by the scientific literature, and solving both is usually a compromise because the adopted solutions can have opposite goals.

Although present at many levels among human or animal societies, no known populational algorithm that is used in optimization modeled the concept of disagreements inbetween individuals. Disagreements bring diversity of opinions inside a group. They are the seed of all great revolutions that changed our societies throughout history and that made us evolve faster, jumping big steps ahead. Most individuals in a population do not disagree with the rules of the society, but a minority is always challenging the established rules, plainly showing disagreement, with more or less emphasis. As time goes by, or in certain circumstances, the minority's voice gets amplified and if their opinions prove better, they end up by revolutioning the general direction. Using this simple observation, this thesis deals with how disagreements can improve the optimization process by simultaneously increasing diversity and the convergence rate.

This thesis introduces and analyzes the concept of disagreements for optimization algorithms. It describes disagreement operators for evolutionary algorithms - represented by genetic algorithms (GAs) - and swarm intelligence algorithms - represented by particle swarm optimization (PSO). Then, it proves empirically that the newly obtained algorithms with disagreements yield superior results for a comprehensive pallette of benchmarks.

The concept of using disagreements in optimization is new and and the results are promising. Because the proposed enhancements using the disagreements concept are made at the core of the algorithms, the area of applicability is as wide and diverse as for the original algorithms.

1.2. Objectives

The main objectives at the time this work was written and the experimental work behind it was elaborated were the following:

a) **A proof of concept** for disagreements in optimization was the general goal of the thesis, a goal that incorporated the other objectives.

b) **A theoretical model** had to be developed in order to introduce disagreements in the context of any evolutionary algorithm.

c) **Specific disagreements operators** had to be designed for PSO and real-valued GAs to specialize the general theoretical model towards swarm intelligence and genetic algorithms.

d) **A testing methodology** had to be set up in order to produce correct data.

e) **A practical performance improvement** had to be observed in order that the concept to be proven useful.

The above enumerated objectives are met by this thesis' contributions that are discussed in detail in the last part of this work, but which can now be summarized as follows:

1) the introduction of the new metaphor of disagreements in swarm intelligence and evolutionary computation in general

2) the establishment of the theoretical foundations for disagreements in evolutionary computation

3) the establishment of a reliable testing methodology to compare disagreements enabled algorithms with their original versions

4) the introduction of the disagreements concept in particle swarm optimization

5) the disagreements concept as a Gaussian-based operator in particle swarm optimization (the 6σ-PSOD operator), and a side-by-side empirical testing and comparison of some PSO variants vs. their enhanced versions that use the 6σ-PSOD

6) a disagreements operator that mitigates swarm stagnation in PSO, the RS-PSOD operator, and a comparison between 6σ-PSOD and RS-PSOD

7) the introduction of the disagreements concept in real-valued genetic algorithms

8) the disagreements concept as a mutation operator in real-valued genetic algorithms, the mutation operator, and a side-by-side empirical testing and comparison of real-valued genetic algorithms versus their enhanced versions that use the mutation

9) overall, a proof of concept for disagreements in swarm intrelligence and evolutionary computation

1.3. Publications

The pillars of this PhD thesis' foundation are represented by the scientific articles that were published and presented by the author at renowned international conferences, 3 out of 5 being published in the prestigious Springer series Lecture Notes in Computer Science (LNCS) and Studies in Computational Intelligence (SCI):

1) Lihu and Ș. Holban. Particle swarm optimization with disagreements. In Y. Tan, Y. Shi, Y. Chai, and G. Wang, editors, *ICSI (1)*, volume 6728 of *Lecture Notes in Computer Science*, pages 46-55. Springer, 2011.

This article is the first one in the series that describe how disagreements can be utilized in evolutionary computation. It introduces the 6σ-PSOD operator and proves empirically using popular benchmarks on some classical PSO configurations that using the new operator yields better results.

2) Lihu and Ș. Holban. Particle swarm optimization with disagreements on stagnation. In Radoslaw K., T.F. Chiu, C.F. Hong, and N. Nguyen, editors, *Semantic Methods for Knowledge Management and Communication*, volume 381 of *Studies in Computational Intelligence*, pages 103-113. Springer, 2011.

Another article on using disagreements for PSO, but this time for stagnation management and the RS-PSOD operator is described as a suitable solution to this issue. Provided experimental results advocate in favor of using disagreements in problems prone to stagnation.

3) Lihu and Ș. Holban. Real-valued genetic algorithms with disagreements. In D. Pelta, N. Krasnogor, D. Dumitrescu, C. Chira, and R. Lung, editors, *NICSO*, volume 387 of *Studies in Computational Intelligence*, pages 333-346. Springer, 2011.

The article shows that disagreements are not limited to PSO or swarm intelligence and that they can be successfully applied to the larger class of evolutionary algorithms as initially provided in the theoretical model. A 6σ-PSOD-like implementation is adapted to real-valued genetic algorithms (6σ-GAD) and tested to demonstrate that disagreements can have a wider usage.

4) Lihu and S. Holban. A study on the minimum number of particles for a simplified particle swarm optimization algorithm. In *Proceedings of the 6th IEEE International Symposium on Applied Computational Intelligence and Informatics (SACI)*, pages 299-303, Timișoara, Romania, May 2011, IEEE.

A study regarding finding a configuration with a minimum number of particles for an à la Pedersen simplified PSO ([76]), PSO-VG. The best found configuration is used for PSO-VG in this thesis.

5) Lihu and S. Holban. Top five most promising algorithms in scheduling. In *Proceedings of the 5th International Symposium on Applied Computational Intelligence and Informatics*, pages 397-404, Timișoara, Romania, May 2009, IEEE.

It is the first published article. It is a literature review and comparison using tests with real-world data of what seemed in 2009 to be the most promising algorithms in scheduling, a subdomain of optimization.

Aside from these publications, there are also two research reports at the base of this work, [56] and [57].

1.4. Outline

Chapter 2, whose content is largely based on the author's annual research reports ([56, 57]), lays the theoretical background. It describes what is optimization, how it is classified, presents effective techniques used in optimization and The No Free Lunch Theorem. Then, there is a brief introduction into genetic algorithms and particle swarm optimization because these algorithms will be modified later to model the disagreements concept. A discussion about the environment used to test and validate the disagreements concept: 10 popular optimization test functions and the software and hardware tools. Conclusions are drawn regarding potential improvement points where disagreements may help in the above-discussed algorithms.

The general theoretical foundation of disagreements in evolutionary computation is described in detail in Chapter 3 and the empirical testing and validation methodology for the new theoretical models is established. The chapter ends with some preliminary conclusions regarding the new theoretical framework.

In Chapter 4 the concept of disagreements is introduced and analyzed in the context of PSO algorithms (both the classic PSO and the à la Pedersen simplified variant). There are two approaches to implement disagreements: the first one is using the 6σ-PSOD operator to model normally distributed disagreements of two kinds: partial and extreme; the second approach is used in PSO to mitigate stagnation when it appears by triggering extreme disagreements among members of the population: the RS-PSOD operator. An empirical analysis is done for both operators and concluding remarks are drawn.

Chapter 5 expands the concept of disagreements for real-valued genetic algorithms in particular and discusses the rationale behind it. Then, the 6σ-GAD operator is provided as a new normally distributed mutation operator. The chapter ends by concluding that the 6σ-GAD operator outperforms other GAs that do not use it on most optimization benchmarks.

At the end, in Chapter 6, conclusions are drawn by presenting a summary of the findings in this thesis. Future work and possible improvements are also discussed.

The bibliography contains all the external resources utilized to write the thesis.

2. BACKGROUND

2.1. Optimization

2.1.1. Definition

Optimization refers to a process of selecting a set of parameters that satisfy a given measure of optimality, with or without constraints. The objective is to pick the best element from a range of alternatives.

Problems that need to be optimized arise from a wide extent of domains, not only from economy or from engineering; they can also arise from the more creative sphere of arts, for example.

Linear systems - that need *linear optimization* - can be solved with a classic technique called linear programming, but non-linear problems cannot be solved easily. This thesis deals with *non-linear optimization* problems.

In most cases, optimization refers to finding the global minimum or maximum of a function *f*, called *the objective function* or *the cost function*.

Before optimization a modeling phase takes place in which *f* is built to reflect the best value in its minimum. In order to address uniformly all the problems, most papers from the scientific literature deal with minimization because maximization is seen as a minimization of *-f*.

Using Bergh's notations from [97], an *unconstrained minimization problem* is defined as:

Definition 1. *Given* $f : \mathrm{R} \rightarrow \mathrm{R}$*, find* $x_G \in \mathrm{R}^n$ *for which* $f(x_G) \leq f(x)$*,* $\forall x \in \mathrm{R}^n$*.*

Finding the minimum or maximum on an interval is guaranteed by *the extreme value theorem* formulated by Weierstrass, which here is stated as in [80]:

Theorem 1. *If a function f(x) is continuous on a closed interval [a,b], then f(x) has both a maximum and a minimum on [a,b]. If f(x) has an extremum on an open interval (a,b), then the extremum occurs at a critical point.*

2.1.2. Classification

When some parameters of the given problem have constraints then *constrained optimization* is needed to solve them. For the sake of simplicity and generality this thesis is focused on unconstrained optimization.

There is also a difference between *local* and *global optimization*. The local optimization is about finding the local minimum specific to a defined zone of the search space, while global optimization means finding the global minimum without been trapped into local minima. In Fig. 2.1, there is a simple visualization of a local (x_L) and a global minimum (x_G) for a polynomial function (also used as an example by [97]).

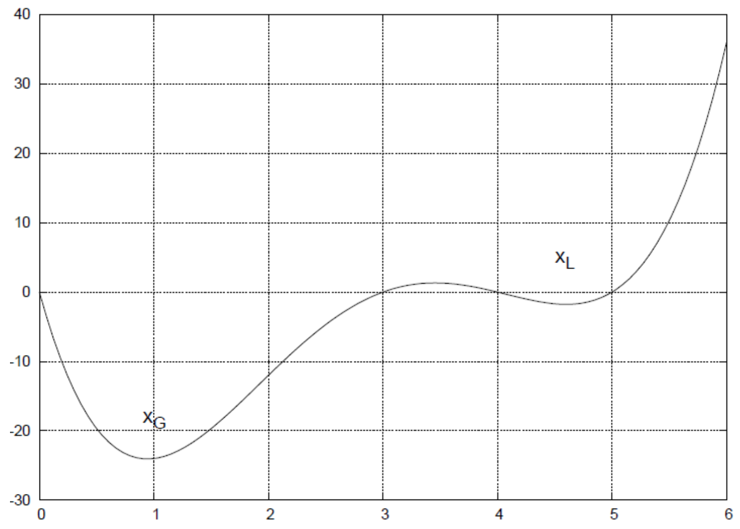


Figure 2.1: The global minimum (x_G) vs. a local minimum (x_L) for the polynomial function

$$f(x) = x^4 - 12x^3 + 47x^2 - 60x\ .$$

With respect to the problem variables and the search space, optimization can be classified as in [74, p. 6] as following:

- *discrete optimization*: when the variables of the objective function assume discrete values (*integer optimization* is the special case for integers)
- *continuous optimization*: when the variables of the objective function assume real values
- *mixed integer optimization*: when the variables of the objective function assume integer and/or real values

Commonly, the objective function does not change over time and there is a single function to be minimized, but when this is not the case, there can be two situations:

- *dynamic optimization*: when there is one or more time-varying objective functions
- *continuous optimization*: when several objective functions need to be concurrently minimized

If the gradient of the optimization problem cannot be known, then the objective function must be optimized as *black box*. This is called *black box optimization*. In addition, optimization algorithms that do not rely on the problem's gradient are referred as "black box", "derivative-free" or "direct search" methods. They are like a blindfolded hiker who wants to find the lowest point in a landscape having access only to the information provided by an altimeter.

2.1.3. The Curse of Dimensionality

Because there can be a large number of candidate solutions to a problem, it is not feasible to try all of them. This means that there is no guarantee that the global optimum is found, but depending on their quality compared to the optimum, there are cases when suboptimal solutions are acceptable. Worse, adding a new dimension to the problem (e.g. an extra parameter) grows exponentially the number of candidate solutions. This is called the *curse of dimensionality*, after Bellman [11, 12]. Therefore, it is important to design algorithms that perform well also when adding new dimensions. Ideally, a new optimization method should have linear time-complexity *O(n)* in the dimensionality *n* of the problem. Not to mention that any optimization algorithm should find the optimum no matter what was its starting position.

2.1.4. Methods

There is a multitude of approaches when it comes to solve an optimization problem and most of them are iterative algorithms. Depending on the problem, there are cases when *deterministic methods* are suitable, like *branch-and-bound methods* ([71]) or *interval optimization* ([42]). Other times, *stochastic optimization techniques* might do better, like *simulated annealing* ([52, 96]) or *Monte-Carlo methods* ([14, 65]). For scheduling problems, that represent a subclass of optimization problems, some of the best methods are presented in one of the articles published by this paper's author, in [59]. Nevertheless, in most scenarios *metaheuristics* do better, thus they are suitable for optimization in worst conditions (black-box optimization). Some of these metaheuristics are listed below:

- evolutionary algorithms (evolutionary strategies, genetic algorithms, etc.)
- swarm intelligence algorithms (particle swarm optimization, ant colony optimization, bee colony optimization, etc.)
- differential evolution ([91])
- memetic algorithms ([68])

This thesis is particularly dealing with particle swarm optimization and genetic algorithms, but as it will be shown later, the new principles that will be introduced throughout this work can be easily applied to other metaheuristics as well.

2.1.5. Convergence in Search Space

Definition 2. *The convergence of an algorithm* a *in the search space* H^n*, in which there is a space measure* $\|\cdot\|$*, is defined as:*

$$\lim_{i \to \infty} \|x_i - x_G\| = 0, \tag{2.1}$$

where x_G is the global minimum and x_i is the best solution at some iteration *i*.

Because an algorithm's number of iterations is finite, a relaxed criterion for convergence is defined as follows:

Definition 3. *Given a space measure $\|\cdot\|$ and a positive integer k, the ε-accurate convergence (ε>0) of an iterative algorithm a_{it} in the search space H^n is defined as:*

$$\|x_i - x_G\| \le \varepsilon, \forall i \le k, \tag{2.2}$$

where x_G is the global minimum and x_i is the best solution at some iteration *i*.

If the global optimum $f(x_G)$ is *a priori* known - and this is the case of the benchmark functions - then the convergence is achieved when the following relation holds true:

$$\|f(x_i) - f(x_G)\| \le \varepsilon. \tag{23}$$

2.1.6. The "No Free Lunch" Theorem

Wolpert and Macready's "No Free Lunch Theorem" (NFL) applies to most algorithms in search and optimization ([98, 99]). It states that the performance of any two algorithms averaged on the set of all possible objective functions in a finite search space is equivalent. Moreover, their performance is even comparable to a random search.

Fortunately, NFL is not valid in all real-world situations. There are finite subsets of the search space in which some algorithms perform much better than others and the incidence of these situations are quite high in practice. Later, Wolpert and Macready proved the existence of the co-evolutionary free lunches in self-play problems ([100]).

From NFL one must understand that there is no overall superior algorithm. Better algorithms can be designed only for particular problems, therefore in order to solve more efficiently an objective function one must have prior knowledge on which method performs better in that case.

2.2. Genetic Algorithms

2.2.1. Evolutionary Computation and GAs

Genetic algorithms (GAs) belong to the greater category of *evolutionary computation* (EC) methods. EC is an umbrella for all population-based meta-heuristics that mimic Darwinian evolution ([26]) to iteratively find the optimal solution. Specifically, the potential solutions, which are represented as individuals are improved systematically through evolution towards the optimum. Friedberg and Bremermann applied for the first time in the fifties the Darwinian concepts in optimization ([21, 36]). Genetic algorithms are one of the most popular optimization methods and they were introduced by Holland in the mid-60's ([44, 45]).

The notion of population representing solutions is central not only to EC, but to the majority of other metaheuristics that do not fall into this category.

Definition 4. *The population of μ individuals at any iteration t that act as potential solutions in the hyperspace of solutions H^n is:*

$$P(t) = \{x_1(t), x_2(t), \ldots, x_i(t), \ldots x_\mu(t)\},\ x_i(t) \in H^n. \tag{2.4}$$

The aforementioned population of individuals representing evolving solutions obeys the rules from nature. Individuals are competing with each other and only the fittest survive, thus are selected to reproduce. During reproduction, the transfer of genetic material might be faulty; there is a slight probability of mutations.

Canonically GAs represent their solutions in a binary form, therefore a mapping function is needed to translate the *genotype* (the binary representation of the solution) into the *phenotype* (the actual form of the solution, i.e. a real-valued representation) and vice versa. With the advent of real-valued GAs ([29, 35]), this disadvantage disappeared.

2.2.2. A General Evolutionary Framework

A general evolutionary framework that characterizes all evolutionary algorithms (EAs), and GAs in particular, was defined by Bäck et al. in [7] and was slightly improved by Bergh in [97]:

Let f be the fitness function that measures the quality of a solution - the objective function - and let $F(t)=\{f(x_1(t)), f(x_2(t)),\ldots, f(x_i(t)),\ldots,f(x_\mu(t))\}$ be the fitness of the whole population. Given the strategy parameters f, μ (the parent population), λ (the offspring population), Θ_s (the probability of selection), Θ_r (the probability of recombination) and Θ_m (the probability of mutation), the general evolutionary framework is defined as in Pseudocode 1.

Pseudocode 1. Bäck's general evolutionary framework.

```
t ← 0
P(t) ← intialise(μ)
F(t) ← evaluate(P(t), μ)
Repeat
    P'(t) ← recombine(P(t), $\Theta_r$)
    P''(t) ← mutate(P'(t), $\Theta_m$)
    F(t) ← evaluate(P''(t), λ)
    P(t+1) ← select(P''(t), F(t), μ, $\Theta_s$)
    t ← t + 1
Until a termination criterion is met
```

2.2.3. Initialization

Population initialization is the first action taken by an EA; it can consist in generating an uniform distribution of the initial population across the search space done either in a deterministic manner when possible - e.g.: spreading the individuals across the nodes of a grid generated on the search space, or stochastically by using an uniform random distribution. A popular initialization technique is the nonlinear simplex method from [70]. More sophisticated intialization schemes can also be employed, but it is generally accepted by researchers that a reliable algorithm should perform well with the uniform approach and should not use the initialization step to gain competitive advantages over other similar algorithms.

2.2.4. Selection

Selection is the process of picking the best individuals according to the fitness function to form a group of parents that will later generate offspring. Two most discussed in scientific literature types of selection are *tournament selection* and *roulette-wheel selection*.

Tournament selection means running a series of contests between a randomly picked set of population members and adding the winners into the parents group. Formally, this can be formulated as in [74]:

Definition 5. *Let P be a population consisting of μ individuals, m be a fixed integer from the set {2,3,…,N}, and k be the number of parents to be selected. Tournament selection can be described with the Pseudocode 2.*

Pseudocode 2. Tournament selection.

```
Do (i = 1..k)
    Choose randomly m individuals from the population P.
    Select one among the m individuals.
    Add the selected individual into the parent pool.
End Do
```

Tournament selection takes two forms: the *stochastic* form and the *deterministic* form. In the stochastic variant those *m* individuals are probabilistically ranked. The best individual has the probability *p* to be selected, the second best has the probability of selection $p(1-p)$, and the last one $p(1-p)^{m-1}$. In the deterministic variant the best individual is selected; in this way *elitism* is used.

When using *roulette-wheel selection* less fit individuals can enter the parents pool and then to each one from the set of *m* individuals the following probability of being picked is assigned:

$$p_i = \frac{f_i}{\sum_{j=1}^{m} f_j}. \tag{2.5}$$

In order to resemble a roulette wheel where each individual occupies a region p_i, an individual holding index $k \in \{1,2..m\}$ is picked such that:

$$\sum_{i=0}^{k-1} p_i \le q < \sum_{i=0}^{k} p_i, \tag{2.6}$$

where $q \sim U(0,1)$ and it is assumed that $p_0 = 0$.

2.2.5. Recombination

The *recombination* process, also called *crossover* for GAs, controlled by the parameter Θ_r, is referring to the process of mixing two parents into giving birth to two new offspring. Because genetic algorithms primarily rely on recombination, the probability rate of crossover is high (around 70%).

Traditionally, since GAs use a binary representation, the recombination takes place on the composing bits of parents' genotype. The simplest case, the *1-point crossover* is defined below, but there are also *multi-point crossover* operators that are defined similarly:

Definition 6. *Given* $p_1 = \{p_{11}, p_{12}, \dots p_{1n}\}$ *and* $p_2 = \{p_{21}, p_{22} \dots p_{2n}\}$ *two randomly chosen parents of length* n *and a crossover point* $k \in \{1,2,\dots n-1\}$*, then the resulting offspring are:* $o_1 = \{p_{11}, p_{12}, \dots p_{1k}, p_{2(k+1)}, \dots p_{2n}\}$ *and* $o_2 = \{p_{21}, p_{22}, \dots p_{2k}, p_{1(k+1)}, \dots p_{1n}\}$*.*

In order to offer a simple vizualisation, consider a parent's genotype be represented by "⊗" and the other one's genotype be notated with "⊕". A custom 3-point crossover is exemplified below:

first parent:	⊗⊗⊗ | ⊗⊗⊗⊗⊗ | ⊗⊗⊗⊗ | ⊗⊗
second parent:	⊕⊕⊕ | ⊕⊕⊕⊕⊕ | ⊕⊕⊕⊕ | ⊕⊕
	– –
first offspring:	⊗⊗⊗ | ⊕⊕⊕⊕⊕ | ⊗⊗⊗⊗ | ⊕⊕
second offspring:	⊕⊕⊕ | ⊗⊗⊗⊗⊗ | ⊕⊕⊕⊕ | ⊗⊗

There are many other variants of binary crossover operators, like *uniform crossover* (for each offspring's new bit selecting with a chance of 50% from each pair of parents' bits) or *cut-and-splice crossover* (offspring with different dimensions resulted from asymmetrical crossover points on parents). However, with the introduction of the real-valued GAs the arithmetic crossover operators were invented. More information about crossover operators can be found in [28].

The simplest arithmetic crossover operator, taken from [97], is defined as follows:

Definition 7. *Given* $p_1(t)$ *and* $p_2(t)$ *as two parents, then the two children can be obtained as follows:*

$$\begin{aligned} o_1(t+1) &= r_1 p_1(t) + (1.0 - r_1) p_2(t), \\ o_2(t+1) &= r_1 p_2(t) + (1.0 - r_1) p_1(t), \end{aligned} \tag{2.7}$$

where $r_1 \sim U(0,1)$ is an uniformly distributed random variable.

Nowadays, one of the most popular arithmetic crossover operators, the one that was used in the experimental part that supports this thesis, is the BLX-α blend crossover that supports multiple offspring. Here it is defined as in [35]:

Definition 8. *Given two parents* $p_1(t)$ *and* $p_2(t)$ *and a control parameter* α*, the ith gene of an offspring h is defined by:*

$$\begin{aligned} h_i &= \mathrm{U}(g_{min} - I * \alpha, g_{max} + I * \alpha), \\ g_{max} &= max(p_{1i}, p_{2i}), g_{min} = min(p_{1i}, p_{2i}), \\ I &= g_{max} - g_{min}. \end{aligned} \tag{2.8}$$

2.2.6. Mutation

The *mutation* perturbs the genotype of one given individual according to the *mutation rate* Θ_m. It promotes diversity because it abruptly changes the genome. Mutation is a "second-class citizen" in GAs, therefore it has a small probability rate (usually 10%).

In binary representations, mutation consists in flipping a randomly picked bit from an offspring with a given probability rate. In real-valued representations, mutation is affecting some vectorial components of the solutions by changing them with some randomly generated numbers inside a predefined range.

A commonly used mutation for real-valued genetic algorithms is the Gaussian distributed mutation, defined as in [28]:

$$y_k(t+1) = o_k(t+1) + \mathrm{N}(0, \sigma_i), \tag{2.9}$$

where o_k is k-th offspring's vector component affected by mutation, σ_i is a user defined variable and y_k is the result of the operation.

A lot of the more elaborate arithmetic mutations are based on Mühlenbein's mutation ([69]), which is described by the following equation:

$$y_k(t+1) = o_k(t+1) \pm rang_k \cdot \gamma, \tag{2.10}$$

where $rang_k = 0.1 \cdot (b_k - a_k)$ is the mutation rate, b_k and a_k are the maximum upper and lower ranges, the signs are chosen randomly with a 0.5 probability and:

$$\gamma = \sum_{i=0}^{15} a_i 2^{-k}, \tag{2.11}$$

where $a_i \in \{0,1\}$ that is randomly generated with $p(a_i = 1) = 1/16$. This special kind of operator generates mutation gradual neighborhoods that do not exceed the given range limit ($b_k - a_k$). More information can be found in [28] and [43].

This section was a short overview on genetic algorithms. For a more detailed description and a deeper analysis on the above presented topics, the reader is suggested to consult Goldberg's book, [40].

2.3. Particle Swarm Optimization

2.3.1. Swarm Intelligence

Particle Swarm Optimization (PSO) is the most prominent optimization technique belonging to the larger category of the *swarm intelligence*.

As stated in [74, p.16], "swarm intelligence is a branch of *artificial intelligence* (AI) that studies the collective behavior and emergent properties of complex, self-organized, decentralized systems with social structure. Such systems consist of simple interacting agents organized in small societies (swarms). Although each agent has a very limited action space and there is no central control, the aggregated behavior of the whole swarm exhibits traits of intelligence, i.e., an ability to react to environmental changes and decision-making capacities."

Bird flocking, fish schooling, animal herding, bee or ant colonies, as well as human interactions stood as the main inspiration sources for swarm intelligence. Early related work first appeared in 1989 in [13], and soon several representative algorithms for this field developed: *particle swarm optimization* ([51]), *ant colony optimization* ([33, 34]), *stochastic diffusion search* ([15, 16, 19]) and *artificial bee colony optimization* ([48]).

Five basic abilities are exhibited in swarm intelligence algorithms, as stated in [66]:

Adaptability: behavioral changes can occur under external factors
Proximity: systems can perform space and time computations
Quality: the capability to respond to environmental quality factors
Diverse response: the faculty of producing several different responses
Stability: the capacity of retaining robust behaviors under soft environmental changes

PSO is a very competitive optimization algorithm initially introduced by Kennedy and Eberhart in [51]. It was inspired from an older algorithm that simulated a bird flock (described in [81]). It is representative to the "social mind" metaphor by simulating the social behavior of groups from the animal kingdom, such as fish or birds. Because PSO does not need any gradient information, it can successfully be used in black-box optimization.

2.3.2. Algorithm Description

In nonprofessional's words, PSO can be described as a population of particles that fly in the search hyperspace of the potential solutions. Until a termination criterion is met, each particle is guiding its flight based on its own experience and the experience of the group it belongs to. All particles are social, they belong to a group, hold positions and have velocities; they remember their best position so far and they know instantly the best position in their group.

The PSO's updating principle is very simple and is made of two components: at each iteration, each particle updates its position towards its personal best - *the cognitive component* - and its group's best - *the social component*.

PSO consists in a social network of particles searching for the overall optimum. They interact and exchange information inside their groups while flying through the search space.

Next, for describing PSO the notation from [97] will be used:

f - is the function to be minimized,

n - is the dimension of the solution hyperspace H^n,

s - is the number of particles in the swarm,

i - is the index of a particle, such that $i \in \overline{1,s}$.

Each particle i deals with the following variables:

x$_i$ - is its current position,

v$_i$ - is its current velocity,

y$_i$ - is its current best position,

$\hat{y}_i$ - is the neighborhood's best position.

Definition 9. *The neighborhood with a size* l *of a particle* i *in the particle swarm optimization algorithm is defined as:*

$$N_i = \{y_{i-l}(t), y_{i-l-1}(t) \ldots, y_{i-1}(t), y_i(t), y_{i+1}(t), \ldots, y_{i+l-1}(t), y_{i+l}(t)\}. \quad (2.12)$$

Particle Swarm Optimization is an algorithm that consists of three phases:

Initialization. The particles' positions are set uniformly in the search space:

$$x_{ij} = \dot{x}, \dot{x} \sim \mathrm{U}(-x_{max}, +x_{max}), i \in \overline{1,s}, j \in \overline{1,n}. \quad (2.13)$$

Velocities are set to 0 or initialized using the following rule:

$$v_{ij} = \dot{v}, \dot{v} \sim \mathrm{U}(-v_{max}, +v_{max}), i \in \overline{1,s}, j \in \overline{1,n},$$
$$v_{max} = k \times x_{max}, k \in [0.1, 1.0]. \quad (2.14)$$

Then, the best positions are updated using eq. (17) and (18).

Iterations. In this phase, the velocities are updated as follows:

$$v_{ij}(t+1) = wv_{ij}(t) + c_1 r_{1j}(t)[y_{ij}(t) - x_{ij}(t)] + c_2 r_{2j}(t)[\hat{y}_{ij}(t) - x_{ij}(t)], \quad (2.15)$$

where c_1 is the personal coefficient, c_2 is the social coefficient, $c_1, c_2 \in (0,2]$. r_1 and r_2 are random vectors, such that $r_1, r_2 \sim U(0,1)$.

The first term of (2.15) is the previous velocity influenced by an inertial weight w. The second term is the personal component that makes the particle move toward its best personal position so far, and the third term makes the particle to turn to neighborhood's best position found so far.

The positions are updated by adding the result from eq. (2.15) to the previous position:

$$x_i(t+1) = x_i(t) + v_i(t+1). \quad (2.16)$$

At the end of each iteration, y_i and $\hat{y}_i$ are updated using the formulae:

$$y_i(t+1) = \begin{cases} y_i(t) & \text{if } f(x_i(t+1)) \geq f(y_i(t)) \\ x_i(t+1) & \text{if } f(x_i(t+1)) < f(y_i(t)) \end{cases}. \quad (2.17)$$

$$\hat{y}_i(t+1) \in N_i \mid f(\hat{y}_i)(t+1)) = min\{f(a)\}, \forall a \in N_i. \quad (2.18)$$

It must be added that if a velocity or position exceeds its range, then *clamping* is employed. This assures that values stay within the desired values.

Termination. It occurs when a given criterion is met (a number of fitness calls or iterations, stagnation, etc.).

2.3.3. Pseudocode

A typical PSO procedure that describes the above long definition can be written as in Pseudocode 3.

Pseudocode 3. Classic PSO procedure.

```
Initialize an n-dimensional swarm S of s particles.
$t \leftarrow 0$
Repeat
    Do (i = 1..s)
      Update $y_i$ using equation (2.17)
      If $y_i < \hat{y}_i$ Then // Update the neighborhood's best if the case
            $\hat{y}_i \leftarrow y_i$
      Update the velocity using eq. (2.15).
      Update position using eq. (2.16).
      End Do
      $t \leftarrow t+1$
Until a termination criterion is met.
```

2.3.4. Flavors

PSO comes in two flavors: the *gbest* and the *lbest* variants.

In the *gbest* variant there is only one big neighborhood that contains all particles and they are fully connected. It was actually the PSO that was initially described by its inventors, Kennedy and Eberhart. In this model all particles know a global best, $\hat{y}$. In this case, eq. (18) is replaced by the following equation:

$$\hat{y}(t) \in \{y_0(t), y_1(t), \ldots, y_s(t) \mid f(\hat{y}(t))\} = \min\{f(y_0(t)), f(y_1(t)), \ldots, f(y_s(t))\}. \quad (2.19)$$

The *lbest* variant has been described before in this section and in this form a particle communicates only inside its neighborhood. The *lbest* model has no spatial relationship between particles because it would be computationally expensive. The size of a particle's neighborhood spans across l indexes of particles.

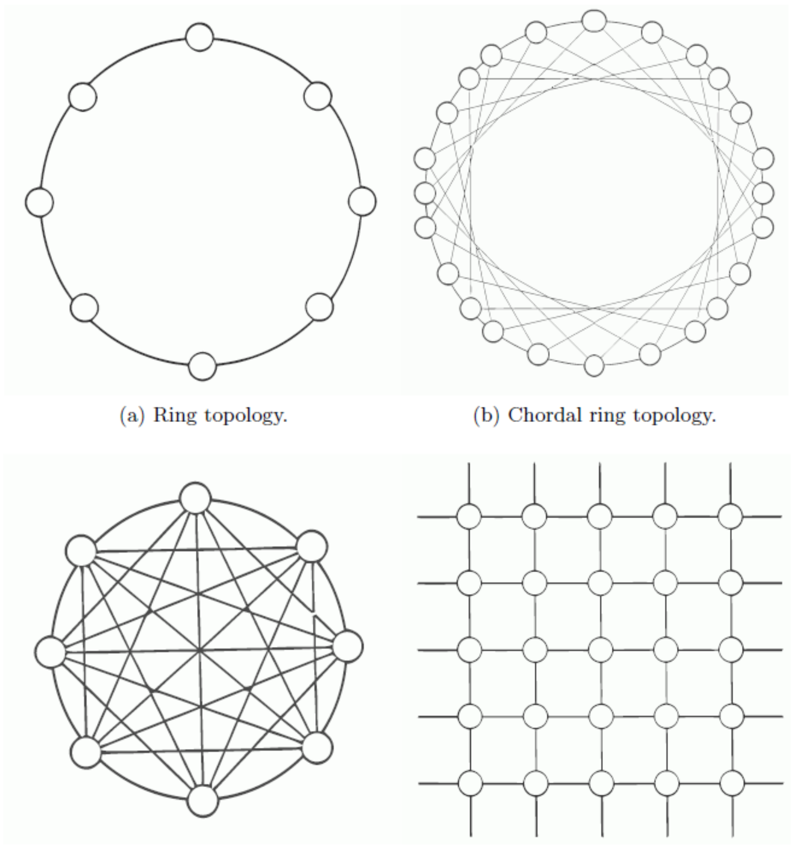
It can be noticed that the *gbest* variant is a special case of the *lbest* variant, with $l = s$. A mix of both flavors can be found in [73] under the name *Unified PSO*.

2.3.5. Topologies

The way particles communicate in their social network determines the *network topology*. There are various network topologies that define "small worlds" for PSO particles in the scientific literature, but only the ones most used in the experimental practice are given below:

- **Ring topology** (see Fig. 2.2a): describes a *lbest* model of PSO where $l = 1$, such that a particle i can exchange information with the particles $i-1$ and $i+1$; the slow informational exchange rate gives the algorithm the ability to explore various regions of the search space.

- **Chordal ring** (see Fig. 2.2b): is an extension to the standard ring topology by adding extra connections between all pairs that are some index-based distance apart in order to increase the information flow and the convergence speed.
- **Fully connected topology** (see Fig. 2.2c): is the *gbest* model where all particles exchange information; this can lead to premature convergence because it is possible that not all the search space is explored when a very good local best is found early.
- **Grid topology** (see Fig. 2.2d): also called Von Neumann topology, is an arrangement in which particles communicate in 4 cardinal directions (N, E, S, W); it is recommended by Kennedy in [50] as the best performing topology in his tests and it is used as the default topology for the experiments with PSO algorithms provided in this work.


(a) Ring topology. (b) Chordal ring topology.

(c) Fully connected topology. (d) Grid topology.

Figure 2.2: PSO topologies.

2.3.6. Inertial Weight

The usage of the *lbest* variant and topologies can bring diversity in the swarm and can enhance a thorough local exploration, but still the swarm needs to converge to a satisfactory solution in a reasonable amount of time. Also, another problem was that of the *swarm explosion*, with particles exceeding by position the search bounds; this was partially solved by clamping to the maximum range. A better solution was the introduction by Shi and Eberhart in [86] of the inertia weight, *w* into the updating principle of PSO in eq. (2.15) in order to boost the refinement process of promising solutions; it is worthy to note that the original PSO developed by Kennedy and Eberhart was not containing the inertia weight.

Good experimental results were obtained for linearly decreasing values of *w* between *[0.9,0.4]* in [88]. The study from [87] found best results for *w=0.8*. Successful simulated annealing-like schemes were proposed also, like the one in [38] and a fuzzy inertia weight in [89].

2.3.7. Constriction Factor

Ozcan and Mohan studied the oscillatory properties of the particles' trajectories in PSO in [72]. Later, Clerc and Kennedy, after a thorough investigation, published the *Standard PSO* variant in [23] that lack the inertia weight, but has a constriction factor instead, therefore eq. (15) was replaced by:

$$v_{ij}(t+1) = \chi[v_{ij}(t) + c_1 r_{1j}(t)[y_{ij}(t) - x_{ij}(t)] + c_2 r_{2j}(t)[\hat{y}_j(t) - x_{ij}(t)]], \qquad (2.20)$$

where χ is called the *constriction coefficient* or *constriction factor*. Although algebraically equivalent to the inertia weight model, this variant is known for its mathematical properties that imply the following selection of the parameters:

$$\chi = \frac{2}{2 - \varphi - \sqrt{\varphi^2 - 4}}, \qquad (2.21)$$

where $\varphi = c_1 + c_2, \varphi > 4$.

The configuration with $\chi=0.729$ and $c_1=c_2=2.05$ is considered the standard configuration for this type of PSO, that also will be used in benchmarks throughout this paper.

2.3.8. Simplifications

The idea to simplify the PSO in order to obtain a more robust algorithm, better to tune, to control and document for multiple optimization problems is not new. It first appeared as the "social-only PSO" in [49] and was later developed by Pedersen in [76]. A study related to the optimal number of particles in such algorithms can be found in one of this thesis author's publications, in [62].

Simplification consists in leaving out one of the components from the updating principle. Pedersen's PSO-VG from [75] (the name states it preserves the previous velocity and the global best to orient itself) has replaced the velocities update formula with:

$$v_{ij}(t+1) = w v_{ij}(t) + c_2 r_{2j}(t)[\hat{y}_j(t) - x_{ij}(t)]. \qquad (2.22)$$

In order to successfully tackle different situations, adaptation schemes can be employed during the optimization process, but this implies a higher operational and computational cost. In [75] Pedersen concludes that it is better to use simplification than adaptation because his results on benchmarks were not too different in both cases.

2.4. Test Functions and Tools

2.4.1. The Need for Benchmarks

It is hard to tell how a new algorithm behaves on a set of real-world problems. In order to prove the quality of an optimization method in real-world scenarios and to have the possibility to compare its performance with other techniques there should be a standard way to do it.

In order to asses a large palette of situations, benchmark functions with various degrees of difficulty were designed to test optimization algorithms and to draw conclusions regarding the convergence speed and the optimality. A set of benchmark functions was first proposed by De Jong in [47] and since then more sophisticated test functions were developed.

Seen in 3D, most of these functions look like relief forms - mountains, valleys, hills etc. Usually, they possess many local minima in order to provide a *high multi-modal environment* in order to rigorously simulate unpredictable real-world situations.

There is no precise way to correctly assess the performance of a black-box stochastic optimization algorithm over a set of problems other than running it and observing the results. Only *empirical analysis* can be effectively used. This happens mainly due to the stochastic nature of the algorithms. Their behavior cannot be exactly predicted by a fixed theoretical model.

2.4.2. Benchmark Biases

Standard test functions are designed in such a way to mimic complex real-world situations and not to give advantage to an algorithm over another. The benchmarks should not be biased. The list with the main biases that can appear in benchmark functions, taken from in [64], is given below along with comments on how they interact with some algorithms.

Initialization bias. Population-based algorithms like PSO do a uniform initialization. On a spherical test function, this means that some individuals could already have found the best solution.

Axial and directional bias. According to [25], binary GAs exhibit better performance on functions with axial biases.

Decomposability/separability. In [94], separability is defined as:

Definition 10 *A function f(x) is separable iff*

$$\arg\min_{x_1,\ldots x_n} f(x_1,\ldots x_n) = \left(\arg\min_{x_1} f(x_1,\ldots),\ldots,\arg\min_{x_n} f(\ldots x_n)\right). \qquad (2.23)$$

[25] has demonstrated that GAs perform better on separable functions than on non-separable.

Rotational invariance. Ackley's function has a long funnel with rotational invariance. The population-based algorithms can push their individuals into that swirl and find easier the best solution.

Regularity. Learning-based methods can exploit found regularities.

Scale bias. At a smaller scale, functions like Rastrigin loose details and transform themselves into simpler functions.

In order to make a test function more difficult, one may try to shift it with an offset or to rotate it. There is no perfect benchmark and that is why there are so many. However, there are some very frequently used ones in the scientific literature and those will be described in the next section and used in tests performed throughout this thesis.

2.4.3. Test Functions

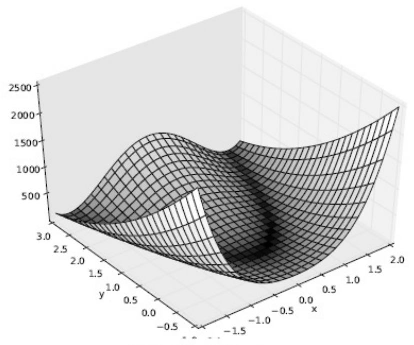
Ten popular test functions were selected from the scientific literature and their characteristics are presented as in [95]. They are renamed with *LF* for a more convenient use. Their graphs are provided in 3D (2 dimensions, x_1 and x_2, plus the fitness function $f(x_1,x_2)$. The shifted functions and their graphs are picked from CEC 2005's list from [92].

1. **Generalized Rosenbrock**

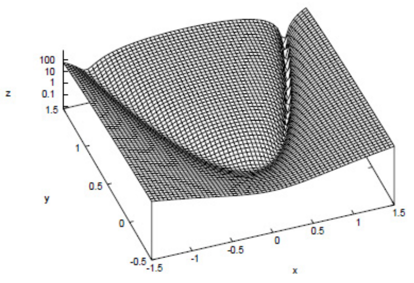
$$LF_1(X) = \sum_{i=1}^{n-1}(100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2),\ X \in [-5,5]^n. \quad (2.24)$$

Global optimum: $X_G = (1,\dots 1),\ LF_1(X_G) = 0$.

Comments: The Generalized Rosenbrock function (see Fig. 2.3, generated with code from [31]), introduced in [82], is an unimodal non-separable function that has a wide flat plateau in which algorithms may fail to find the global optimum that resides in a very thin and steep valley inside the aforementioned plateau (see Fig. 2.3b, generated with code from [93]).

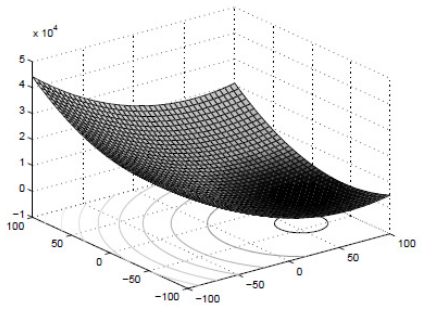


(a) Overall view.

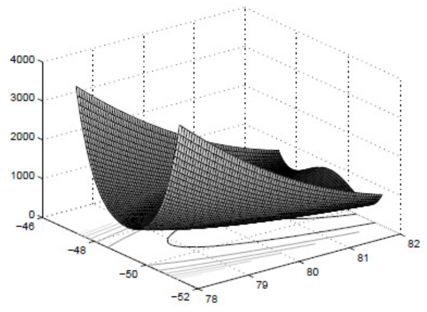


(b) The steep valley.

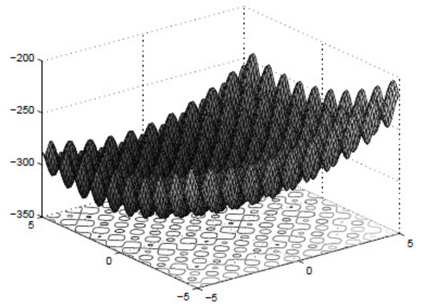
Figure 2.3: Generalized Rosenbrock function.



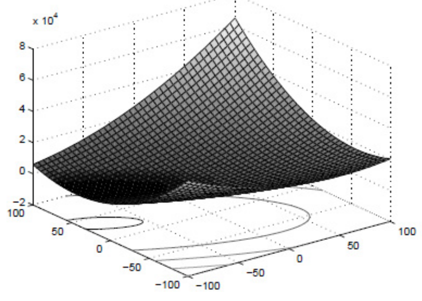
(a) Shifted Sphere.



(b) Shifted Rosenbrock.



(c) Shifted Rastrigin.



(d) Shifted Schwefel.

Figure 2.4: CEC 2005 shifted functions.

2. **Shifted Sphere**

$$LF_2(X) = \sum_{i=1}^{n} z_i^2 + f_{bias_2}, \; Z = X - O, \; X \in [-100,100]^n, \tag{2.25}$$

where $O = [o_1, \ldots o_n]$ is the shifted global optimum.

Global optimum: $X_G = O, LF_2(X_G) = f_{bias_2} = -450$.

Comments: The Shifted Sphere (see Fig. 2.4a) function is an unimodal, shifted, separable and scalable function derived from its simple and unshifted variant. Usually an algorithm fails this test when it contains theoretical mistakes. It is a test for general efficiency.

3. **Shifted Rosenbrock**

$$LF_3(X) = \sum_{i=1}^{n-1}(100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2) + f_{bias_3},$$

$$Z = X - O + 1,\ X \in [-100,100]^n, \tag{2.26}$$

where $O = [o_1, \ldots o_n]$ is the shifted global optimum.

Global optimum: $X_G = O,\ LF_3(X_G) = f_{bias_3} = 390$.

Comments: The Shifted Rosenbrock (see Fig. 2.4b) function is a multi-modal, shifted, non-separable and scalable function derived from Generalized Rosenbrock function. It is harder to solve that its simpler counterpart and retains the capability to provide information on how an algorithm can tackle plateau functions.

4. **Shifted Rastrigin**

$$LF_4(X) = \sum_{i=1}^{n}(z_i^2 - 10\cos(2\pi z_i) + 10) + f_{bias_4},\ Z = X - O,\ X \in [-5,5]^n, \tag{2.27}$$

where $O = [o_1, \ldots o_n]$ is the shifted global optimum.

Global optimum: $X_G = O,\ LF_4(X_G) = f_{bias_4} = -330$.

Comments: The Shifted Rastrigin(see Fig. 2.4c) function is a multi-modal, shifted, separable and scalable function derived from the Rastrigin function. It can check for behavior of convergence in the presence of a huge number of local minima and large basin of attraction.

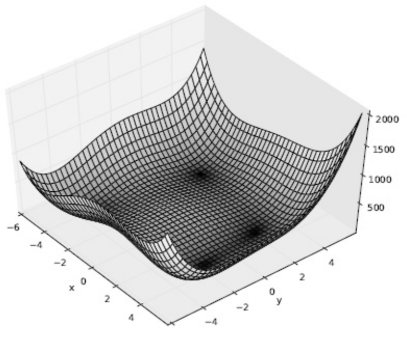
5. **Shifted Schwefel 1.2**

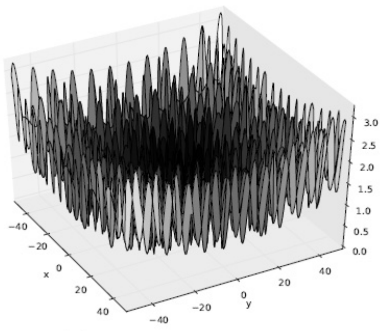
$$LF_5(X) = \sum_{i=1}^{n}\left(\sum_{j=1}^{i} z_j\right)^2 + f_{bias_5},\ Z = X - O,\ X \in [-100,100]^n, \tag{2.28}$$

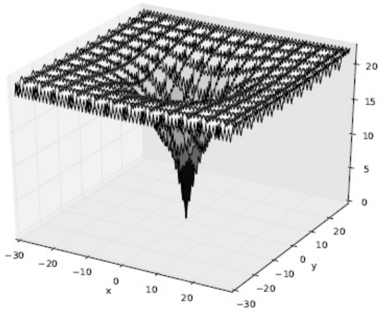
where $O = [o_1, \ldots o_n]$ is the shifted global optimum.

Global optimum: $X_G = O,\ LF_4(X_G) = f_{bias_5} = -450$.

Comments: The Shifted Schwefel 1.2 problem (see Fig. 2.4d) is an unimodal, shifted, non-separable and scalable function derived from the simpler Schwefel 1.2 function. As stated in [22], it can provide a hint on the algorithm's robustness.


(a) Himmelblau function.


(b) Griewank function.


(c) Ackley function.

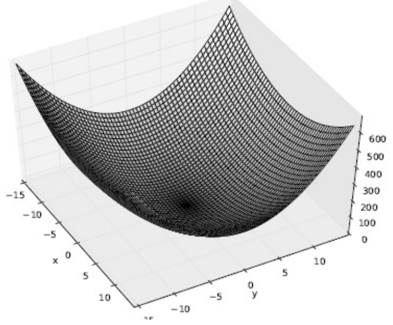

(d) Bohachevsky function.

Figure 2.5: Other multimodal functions.

6. **Himmelblau**

$$LF_6(x_1,x_2) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2,\ x_1,x_2 \in [-6,6]. \tag{2.29}$$

Global optima: One global maximum at $LF_6(-0.270844,-0.923038) = 181.616$ and four identical global minima: $(x_{1min}, x_{2min}) \in \{(3,2),(-2.805118,3.131312),(-3.779310,-3.283196),$ $,(3.584428,-1.848126)\}$ where the function's value is 0.

Comments: A low-dimensional multi-modal plateau function (see Fig. 2.5a, generated with code from [32]).

7. **Griewank**

$$LF_7(X) = \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{s} cos\left(\frac{x_i}{\sqrt{i}}\right) + 1, X \in [-600,600]^n. \tag{2.30}$$

Global optimum: $X_G = (0,\ldots0), LF_7(X_G) = 0$.

Comments: Griewank (see Fig. 2.5b, generated with code from [79]) is a multi-modal, separable and scalable function.

8. **Ackley**

$$LF_8(X) = -20 \cdot exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - exp\left(\frac{1}{n}\sum_{i=1}^{n}cos(2\pi \cdot x_i)\right) +$$

$$+ 20 + e, X \in [-20,20]^n. \quad (2.31)$$

Global optimum: $X_G = (0,\ldots 0), LF_8(X_G) = 0$.

Comments: Ackley (see Fig. 2.5c, generated with code from [79]) is a multimodal, separable and scalable function that is like a long funnel towards the global optimum.

9. **Bohachevsky 1**

$$LF_9(X) = \sum_{i=1}^{n-1}(x_i^2 + x_{i+1}^2 - 0.3\,cos(3\pi x_i) - 0.4\,cos(4\pi x_{i+1}) + 0.7),$$

$$X \in [-100,100]^n. \quad (2.32)$$

Global optimum: $X_G = (0,\ldots 0), LF_9(X_G) = 0$.

Comments: Bohachevsky's first test function (see Fig. 2.5d, generated with code from [79]) is an unimodal, separable and scalable function.

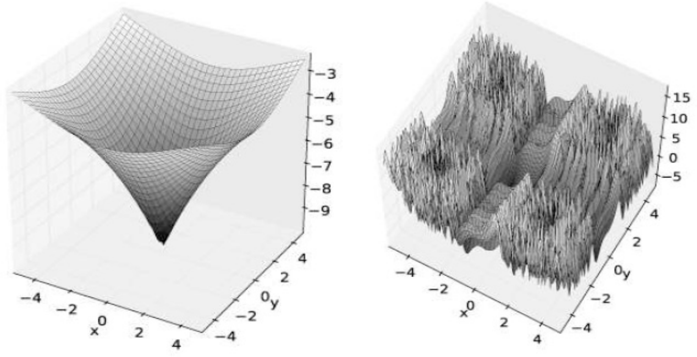
10. **Kursawe**

$$LF_{10a} = \sum_{i=1}^{n-1}(-10\,exp(-0.2\sqrt{x_i^2 + x_{i+1}^2})) \quad (2.33)$$

$$LF_{10b} = \sum_{i=1}^{n-1}(|\,x_i\,|^{0.8} + 5\,sin(x_i)^3) \quad (2.34)$$

Global optimum: $X_G = (0,\ldots 0), LF_{10a}(X_G) = -10, LF_{10b}(X_G) = 0$.

Comments: Kursawe's functions (Fig. 2.6a and 2.6b, both generated with code from [79]) from [55] are useful in multi-objective optimization. Algorithms should optimize both of them simultaneously.



(a) First function. (b) Second function.

Figure 2.6: Kursawe functions.

2.4.4. Tools

In the following subsection, the tools that were used for conducting the experiments within the research associated with this work will be enumerated along with a short presentation line for each one of them:

Hardware Tools

1st PC system: CPU - Intel Pentium i7 2.66 GHz; RAM - 6 GB; running Microsoft Windows 7 Enterprise 64-bit.

2nd PC system: CPU - Intel Pentium T2250 1.73 GHz; RAM - 2 GB; running Microsoft Windows 7 Professional 32-bit.

Software Tools

Java EvA2: An extensive evolutionary framework to implement, compare and test optimization algorithms. The software and its documentation is available at http://www.ra.cs.uni-tuebingen.de/software/EvA2.

IntelliJ IDEA: An IDE to modify the sources of Java EvA2 in order to create new algorithms, available at http://www.jetbrains.com/idea.

Microsoft SQL Server: A relational database server utilized to store experimental data. http://www.microsoft.com/sqlserver/en/us/default.aspx.

SPSS: A computer program used for statistical data analysis and for generating plots. Available at http://www-01.ibm.com/software/analytics/spss/.

gnuplot: A plotting utility, available at http://www.gnuplot.info.

Matlab: A fourth generation language for numerical computing and plotting available at http://www.mathworks.com/products/matlab.

Matplotlib: "... a python 2D plotting library which produces publication quality figures in a variety of hardcopy formats...". Available for download at http://matplotlib.sourceforge.net.

From the above listed set of software tools, Java EvA2 needs special attention. It is a modular evolutionary framework that contains most of the nowadays-evolutionary algorithms. This is very important because the researcher has the possibility to implement her/his own algorithms inside the framework and do a direct comparison in terms of performance with the other provided methods. More, this framework has statistics and plotting abilities and also offers support to solve real-world optimization problems. At the moment this PhD thesis was written, after a thorough evaluation, Java EvA2 was considered by the author as the best available evolutionary framework and the most suitable to perform experiments with. For details on how to use it, the reader is referred to the Java EvA2's website and to a related article regarding this evolutionary framework, [54].

2.5. Concluding Remarks

This chapter summarized the most important aspects of optimization, real-valued genetic algorithms and particle swarm optimization, in order to prepare the reader for the introduction of the *disagreements concept* applied in optimization problems throughout the next chapters.

If people want superior products, efficient use of resources or maximized profits then they should think starting to optimize their processes. It is in the human nature to thrive for better and better and achieve continuous progress.

PSO and GAs are two popular optimization methods that find the best solutions in hard problems without gradient information and this makes them suitable to deal with most optimization problems in design, engineering and economics. However, like all optimization algorithms, it is not guaranteed that they will find the global optimum in all situations. An algorithm should always keep a balance between the *exploration* of new regions and the *exploitation* of the current best solution and the space in its vicinity. For PSO and GAs the exploration is achieved by increasing the population's diversity, thus slowing the convergence, but increasing the probability to find better solutions. On the other hand, the exploitation would increase the convergence rate, but could lead to premature convergence and the algorithm could miss the global optima. Both algorithms have internal basic mechanisms to boost the convergence and to assure a high diversity. To increase the diversity across population, GAs use mutation while PSO uses the *lbest* model with neighborhood topologies. To speed up the convergence, GAs use the crossover operator while PSO adopted the inertia weight or the constriction.

Although most evolutionary algorithms, including swarm intelligence, incorporate both exploration and exploitation components there is still a lot of room for improvement and this is the main purpose of this PhD thesis. It introduces a new metaphor that helps algorithms find better solutions in a shorter period and proves it on the most important optimization test functions. The new idea is presented and tested in the next chapters.

3. DISAGREEMENTS

3.1. Rationale

Disagreements represent a normal social phenomenon between members of a population. A disagreement regarding an issue can appear in any social group at any given time. The history of our civilization and our culture as human beings recorded many cases of individuals or groups of individuals that challenged the norm. In an economics-related study on the dynamics of the disagreements and the opinion fluctuations in a social network, in [2], Acemoglu et al. state that:

"Disagreement among individuals in a society, even on central questions that have been debated for centuries, is the norm; agreement is the rare exception. How can disagreement of this sort persist for so long? Notably, such disagreement is not a consequence of lack of communication or some other factors leading to fixed opinions. Disagreement remains even as individuals communicate and sometimes change their opinions."

Blackwell and Dubbins theorem in [17] about "merging of opinions" or Savage's theorem in [85, p.48] are some classic theories proving that if two individuals observed the same sequence of events then they will agree on *a single world variant* regarding an issue, no matter what were their prior knowledge on the topic. Agreement is settled through *learning*. Recent work done by [3, 4, 8, 9, 37, 41, 90, 103] emphasizes that over a strongly connected network a consensus is typically reached. The same phenomenon can be observed in PSO variants: for the *gbest* model, which has a fully connected topology, the swarm typically converges faster than in the *lbest* case.

There are also other models, such as Axelrod's in [6], in which individuals with similar opinions tend to communicate more often (also like in [30, 53]) and form clusters ([18, 24, 63]), but those models cannot explain the continuous opinion fluctuations in our society, and more than that, they are not useful in swarm intelligence algorithms because they do not promote diversity.

Nonetheless, despite the above-mentioned theories, disagreements are ubiquitously found even in strongly connected networks where the observed sequence of data is the same. This issue is discussed in [1] and the conclusion is that:

"In none of these cases can the disagreements be traced to individuals having access to different histories of observations. It is rather their interpretations that differ."

It is widely accepted that scientists, engineers or economists routinely disagree on the same central issues in their field based on their different interpretations of the same facts. Their disagreements on the same set of data can arise from different importance they give to various variables in the considered problem. Different generations of human beings usually disagree at certain times, like teenagers vs. their parents. The Internet discussion forums are abundant in disagreements on a very large number of topics. The examples can continue in many other areas.

Disagreements exist everywhere there is a society. They are a change promoter and they bring evolution and variety into the social systems. They can be extreme and visible when promoted by people who directly and forcefully challenge the status-quo or they can go unnoticed when there is mild discontent. The rationale for the research behind this PhD thesis is that swarm intelligence algorithms - and by extension all the population-based optimization algorithms - can benefit from the disagreements the same way our real-world society benefits from them. This PhD thesis provides a comprehensive extension to the author's previously published articles that introduced disagreements in the theory of global optimization.

3.2. Foundations

The disagreements concept and its foundations are introduced in this chapter the same way as in the author's previously published works in [60, 61, 58]. In order to experiment with different disagreement types, this chapter provides the fundamental framework for disagreements in the optimization theory using the basic definitions provided below and three point of views: a low level one - how an individual's behaviour is changed and how this is seen from this low level, a medium level point of view - how disagreements are applied on individuals from an iteration and how the update principle is altered, and a high level point of view - from the perspective of creating disagreements-enabled algorithms.

3.2.1. From the Individual's Point of View

The below definition describes the algorithm's update step from an individual and local point of view.

Definition 11. *Given two succesive iterations,* t *and* $t+1$*, from an evolutionary algorithm* E *and a population of* s *individuals,* $P(t)=\{x_1(t), x_2(t), \ldots, x_s(t)\}$*, there is a corresponding vector of update behaviors* $\mathrm{B}(t)=\{\beta_1(x_1,t), \beta_2(x_2,t), \ \ldots, \beta_s(x_s,t)\}$ *that makes the transition from* $P(t)$ *to* $P(t+1)=\{x_1(t+1),\ x_2(t+1), \ldots x_s(t+1)\}$*.*

The update behaviors stem from applying the general updating principle of the algorithm. The update behaviors are the final effects upon the individuals from the population.

Definition 12. *A disagreement is defined as a function* D *that operates on an individual's* x_i *update behavior* β_i *at iteration* t*.*

Definition 13. *Disagreements are defined as the family of functions:*

$$F_D = \{D : R^n \to R^n \mid \forall z \in R^n . D(z) \neq z\}. \tag{3.1}$$

The identity function (no disagreement should happen) is written $\varnothing_D$.

For a simple visualization of the concept, Fig. 3.1 depicts a situation in which there are several "rebels" (pictured in black) among a population.

3.2.2. From the Iteration's Point of View

At the iteration level there is the updating principle that selects individuals and apply different update behavior to them. In disagreements-enabled algorithms there is an apply rule called a "disagreement selector" which alters the general updating principle and decides at each iteration to which individuals a specific type of disagreement is enforced.

Definition 14. *Let ρ be the disagreement selector that decides which disagreement is invoked upon an individual update behavior β_i at iteration t from a given set of disagreements $\Delta_V \in \mathrm{P}(F_D)$:*

$$\rho(\Delta_V, \beta_i, t) = D_j, D_j \in \Delta_V, i \in \overline{1,|\mathrm{B}|}, j \in \overline{1,|\Delta_V|}. \quad (3.2)$$

The original behaviours change when disagreements are injected, therefore $\mathrm{B}(t)$ becomes $\mathrm{B}_D(t) = \{\beta_{D1}(x_1,t), \beta_{D2}(x_2,t), \ldots, \beta_{Ds}(x_s,t)\}$. The general updating principle is not totally replaced because when ρ yields $\varnothing_D$ the original behavior stays in place and no disagreement is invoked. The golden rule is that disagreements should not happen more often than regular updates would, averaged on all iterations of an algorithm run, and they must be linked to the algorithm's context (e.g. social, cultural, etc.) and particularities.

The process of applying a disagreement to an individual of the population P at some iteration t can be seen overly simplified if described by Pseudocode 4:

Pseudocode 4. The insertion procedure of disagreements in an EA.

```
For each update behavior $\beta_i$ from B
    Using $\rho$, apply the appropriate $D_i$ disagreement to $\beta_i$.
End
```

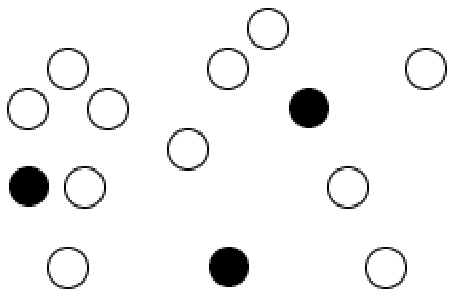


Figure 3.1: Disagreements among individuals of a population.

3.2.3. From the Algorithm's Point of View

The *disagreement injector* that enhances any evolutionary algorithm E_i with disagreements is defined as follows:

Definition 15. *Let E be an EA and ρ a disagreements apply rule. An* ***evolutionary algorithm with disagreements (EAD)****, E_D, is obtained by modifying E's updating principle with the rule ρ, as described by the disagreement injector function Ψ:*

$$\Psi(E,\rho) = E_D. \qquad (3.3)$$

Using the above defined disagreement injector in eq. (3.3) and depending on the given algorithm and the problem to be solved, various disagreement schemes can be designed in order to benefit from the advantages of this concept: both enhanced exploitation and exploration and increased convergence rates without negatively impacting the robustness of the original algorithms, as it will be demonstrated in the following chapters. The apply rule - ρ, is the *de facto* **disagreement operator** for the evolutionary algorithm E.

3.3. Methodology

3.3.1. DACE Fundamentals

Unfortunately, due to their stochastic nature, the behavior and the performance of the evolutionary methods cannot be predicted with great accuracy using an established mathematical model; they can be analyzed empirically using *computational statistics* ([39]), sometimes employing techniques such as *regression analysis* ([20]) and/or *experimental design* ([67, 84]).

In order to do a proper assessment on the impact that disagreements have on evolutionary algorithms, the experiments behind this thesis were crafted using a methodology that is inspired from Bartz-Beielstein et al. in [10], who provided an excellent framework in experimental research that can be applied universally in evolutionary computation because it successfully blends *design and analysis of computer experiments* (DACE) with *design of experiments* (DOE) and with *classification and regression trees* (CART).

In Bartz's usage of DACE, an *algorithm design*, represented by a vector a_d, contains specific characteristics for an algorithm. An optimal design is denoted as a_d^*. A *problem design*, p_d, is a vector describing the particularities of the given problem. A *run* is therefore a mapping between an algorithm design and a problem design, $Y = (a_d, p_d)$. The aim of design and analysis of computer experiments (DACE) is to find the optimum a_d^* through repeated trials, constrained by a predefined number of function evaluations (preferably low).

DACE is useful at interpolating observation points from a massive set of computer experiments. For a stochastic process with an assumed zero mean Z, the dynamic response $Y(x)$ for an input vector x as a realization of the regression model, F, was provided by [83] as a generalization of the classic regression model $Y(x) = \beta x + \varepsilon$:

$$Y(x) = F(\beta, x) + Z(x). \tag{3.4}$$

The covariance for the stochastic process $Z(x)$ is given by the following formula:

$$V(\omega, x) = \rho^2 R(\theta, \omega, x), \tag{3.5}$$

where R, the correlation function, is chosen taking into consideration the actual process. ρ^2 is the process covariance.

Another useful technique within DACE is the *sequential design* in which an initial design $a_d^{(0)}$ is initially generated. Based on the reported data associated with its run on the given problem, it is replaced by another related design, $a_d^{(1)}$. This operation continues until a satisfactory design is obtained. New designs are generated based either on the improvement expectancy, thus betting on promising design points with a good forecast, or on design with a high degree of uncertainty ([84]). Improvement is defined as in [74, p. 65]:

Definition 16. *Let, y_{min}^k, denote the smallest detected function value after k runs of a heuristic global optimization algorithm; $x \in a_d$, be a component of the design; and $y(x)$ be the response of the algorithm, which is a realization of $Y(x)$ defined in eq. (3.4). Then, the improvement of the algorithm is defined as:*

$$\lambda = \begin{cases} y_{min}^k - y(x) & \text{if } y_{min}^k - y(x) > 0 \\ 0 & \text{otherwise} \end{cases} . \tag{3.6}$$

In order to build trustworthy statistics, experiments with stochastic algorithms must be run for a sufficiently large number of times and for a reasonable amount of time, the initialization must be random with different seeds, the test data should be comprehensive, large and diverse enough.

3.3.2. Race Testing

The design of algorithms with disagreements takes into account a large part of DACE methodology, but simplifies the procedures and adapts them to evolutionary and swarm-based methods. Most experiments are conducted as a **race** between a genetic or PSO algorithm with a classical configuration against its disagreements-enabled counterpart. With this derived methodology, called *race testing*, the main focus is on the discovery and analysis of new disagreement operators, in order to improve the best classical configurations so far; improving promising but not best design points in classical algorithms is a secondary goal. The experiments with disagreements in evolutionary computation respected the guidelines of building relevant statistics: random initialization with different seeds, each configuration is run many times, various and well-known benchmarks are used.

Experimentation that leads to the results presented in this PhD thesis consisted of three main phases:

Discovery. In order to study the effects of disagreements, in the first stage there is no need for sequential design; the scientific literature provides classical configurations of algorithms and problems (benchmarks), so first it is needed to test whether the disagreements-enabled methods work *at all* on a small set of benchmarks, with classical settings. Usually a classical algorithm configuration and its disagreements counterpart is tested on 2-3 benchmarks. The exact steps are described in Table 3.1. The termination criterion is based on a large number of function evaluations and on the ε-convergence, which one is accomplished first. Convergence graphs, if the algorithms "ε-converged", or best individual evolution graphs, if not, are analyzed. It mainly implies the theoretical creation of the new disagreements operator. This phase is at the origin of the whole elaboration of the new disagreements algorithms.

Table 3.1: Phase I. Discovery methodology.

Step	**Activity**
Step 1	A new disagreements operator idea.
Step 2	Mathematical description.
Step 3	Disagreements injection into an EA using formula (37).
Step 4	Race specifications:
	- 2-3 benchmark problems,
	- the original algorithm (important factors),
	- the disagreements-enabled algorithm,
	- the termination method,
	- the experimental design,
	- a performance measure.
Step 5	Experimentation.
Step 6	Evaluation and visualization.
Step 7	Statistical data analysis.
Step 8	If:
	- better results, then **go to Phase II.**
	- promising results, then **tweak parameters and go to Step 5.**
	- bad results, then **quit.**

Confirmation. The second stage consists in experimentation on a larger scale by testing more hypotheses. As described by Table 3.2, a set of tests called algorithm races usually take place on all the 10 benchmark functions described in subsection 2.4.3.

Then, sequential design can be used to tune different parameters from the disagreements operators, while still using fixed classical configurations. The purpose of this phase is to study how the injected disagreements act upon algorithms across a large palette of benchmarks and conditions (see Table 3.3).

A later sub-phase using a holistic approach can consist in tuning both the disagreements' operator parameters and the underlying algorithm's parameters dynamically. However, because this work is only a proof of concept for disagreements, the experiments behind it did not follow this path.

Table 3.2: Phase II (a). Confirmation on classical configurations.

Step	Activity
Step 1	Hypothesis.
Step 2	Race specifications:
	- 10 benchmark problems,
	- the original algorithm (important factors),
	- the disagreements-enabled algorithm,
	- the termination method,
	- the experimental design,
	- a performance measure.
Step 3	Experimentation.
Step 4	Evaluation and visualization.
Step 5	Statistical data analysis.
Step 6	Objective interpretation of the results.

Table 3.3: Phase II (b). Sequential design for improving the new disagreements operators.

Step	Activity
Step 1	Hypothesis.
Step 2	Race specifications:
	- one benchmark problem.
	- the original algorithm (important factors),
	- the disagreements-enabled algorithm,
	- the termination method,
	- the experimental design,
	- a performance measure.
Step 3	Experimentation.
Step 4	Evaluation and visualization.
Step 5	Statistical data analysis.
Step 6	Check termination criterion, if satisfied go to Step 8.
Step 7	Tweak parameters and go to Step 3.
Step 8	Acceptance/Rejection of hypothesis.
Step 9	Objective interpretation of the results from Step 8.

Robustness. The last phase of testing is checking for conditions that decrease performance by properly assessing the robustness of the newly obtained algorithms that are enhanced by disagreements. Of course, valid conclusions for Phase III can be drawn from Phase II already and they represent starting points when studying situations in which algorithms begin to encounter difficulties.

The termination criterion that is used to determine when an algorithm stops in any experimental run is based on a predefined number of function evaluations. This approach brings the possibility to draw and interpret the convergence graphs or the best individual graphs easily.

3.3.3. Common Experimental Setup

All conducted experiments in this thesis used the methodology presented in the above Subsection 3.3.2. For brevity, they will not be described systematically as in Tables 3.1, 3.2 and 3.3 throughout this work. Also for brevity, no robustness studies are provided. In this thesis, only the successful configurations after experimentally assessing their robustness are included.

Tests took place for all 10 provided benchmark problems in 2.4.3 in 30 and 50 dimensions. Each experiment's output was measured and averaged over 100 runs. The termination condition for any run was set to 30000 function evaluations.

For PSO, two configurations were used in studies:

SPSO: Maurice Clerc's Standard PSO, a constriction-based PSO with $\chi = 0.729$ and $c_1 = c_2 = 2.05$ ([23]), implemented in Java EvA2.

PSO-VG: an à la Pedersen simplified PSO, a social-only PSO with $w = 0.729$ and $c_2 = 1.49445$.

In both two configurations for PSO, the default is the grid topology with a neighborhood range of 2. Using the conclusions regarding the optimum number of particles in PSO-VG from the author's article [62], experiments took place in both cases for a swarm size of 25 and 50.

For testing GAs, a classical GA was considered in two configurations: with and without elitism. Tournament selection was employed in all cases, with a low number of 50 individuals in the pool. All GAs used a BLX-α crossover with $\alpha = 0.5$ and a probability of 0.7 and a Mühlenbein mutation with a probability rate of 0.1.

In all cases, the disagreements-enabled algorithms preserved the original algorithm's parameters and only their own extra parameters are tweaked or checked for robustness.

For any conducted experiment the mean best fitness and its standard deviation is calculated across the above mentioned 100 runs. Other important output information is provided by the reported convergence ratio - the ratio between the number of successful runs that hit the target with an $\varepsilon = 0.01$ accuracy as in eq. (2.3) and the total number of runs-, the median best fitness value and information related on how many disagreements occurred per iteration.

Averaged graphs with the evolution of the best fitness are drawn to study the convergence behavior across all runs. Worst fitness individuals history graphs are also generated. The evolution of the average distance between the individuals or particles is caught in a separate graph. Values in graphs for best and worst fitness are displayed on a decimal logarithmic scale (on y-axis), while the graphs for average population distance use a linear scale for values (on y-axis). These variables are plotted across all fitness function calls (or evaluations), that are represented on a linear x-axis.

In order to prove the concept of disagreements in every theoretical construction that is provided and for simplicity and brevity, **only relevant testing situations are presented and analyzed in this thesis**. The modified Java EvA2 software that contains the author's additions, namely the 6σ-PSOD, the RS-PSOD and the GAD algorithms, is available for download at https://github.com/andrei-lihu/Eva2-AL.

3.3.4. Concluding Remarks

This chapter provided both a theoretical foundation for disagreements in the context of evolutionary and swarm intelligence algorithms and a methodology to develop and test new disagreements operators for classical algorithms.

The rationale for disagreements is their ubiquity among populations regardless connectivity and information transmission speed. Therefore, needless to say, any population-based algorithm can implement this new metaphor.

A general mathematical model of disagreements and the way they can be injected in EA algorithms was described.

The next issue in focus was how to test the performance of the new concept. After a short introduction into DACE, *the race testing methodology* was depicted in detail and was proposed as the standard way of benchmarking disagreements operators.

4. PARTICLE SWARM OPTIMIZATION WITH DISAGREEMENTS

4.1. Disagreements as A New Social Behavior

4.1.1. Concepts

In PSO, a particle follows a path that is influenced by its personal experience and its social experience. Even in simplifications of the original algorithm the social component is always preserved, therefore PSO is a social algorithm in the first place. The problem with PSO is that in some circumstances its search mechanism can be trapped into local minima. Other times, the algorithm either converges too fast or fails to converge at all. The particles' oscillatory movement, while at the core of PSO's performance over random-search, it can miss some important areas in the search space. Moreover, in the social model in PSO, even if intended to be simple, a particle is always guided by the social norm. Aside from the best particle in the group, most of the time particles are "pure followers" and there is no challenging of the status-quo like in all real-life societies and groups.

By modeling disagreements in PSO, the particles get their own personality. Not all will follow an oscillatory path, therefore there is an increased possibility to explore areas that would have been harder to target before. Each particle can have a various degree of disagreement: it can *partially* agree, or *extremely* disagree with where it should have travelled next in the search space. Under this differentiation, the *partial disagreements* are used to enhance the local exploitation, while the *extreme disagreements* are used for increased exploration.

The disagreements in PSO, as described by the author in [58], are affecting only the social component of the algorithm. Mathematically, a disagreements injector function for PSO, similar to the one from the formula (3.3), must be developed as follows:

If the first term from the updating principle of PSO from eq. (2.15) is replaced with a generic velocity component, denoted with $V(t,i)$, then, the cognitive component with $C(t,x_i,y_i)$ and the social component with $S(t,x_i,\hat{y})$, and finally making the substitution in eq. (2.16), where the position component is replaced with a generic one, $X(t)$, the following generalized updating equation for a particle i at an iteration $t+1$ is obtained:

$$\begin{gathered} X(t+1,x_i) = X(t,x_i) + V(t,i) + C(t,x_i,y_i) + S(t,x_i,\hat{y}_i) + \zeta, \\ C(t,x_i,y_i) \to y_i, S(t,x_i,\hat{y}_i) \to \hat{y}_i, \forall i \in \overline{1,s}, \end{gathered} \tag{4.1}$$

where $C(t,x_i,y_i) \to y_i$ is read as "the cognitive component tends to y_i" and $S(t,x_i,\hat{y}) \to \hat{y}$ - "the social component tends to $\hat{y}$". ζ, which is usually equal to 0, can accommodate any other PSO that has more components. **The change to the social component must be made with the intent of not following the leader (best particle) of the group.**

In PSO, it can be easily noticed that the update behavior for any particle i, β_i, is represented by the right term of the above eq. (4.1):

$$\beta_i(x_i,t) = X(t,x_i) + V(t,i) + C(t,x_i,y_i) + S(t,x_i,\hat{y}_i) + \zeta, \forall i \in \overline{1,s}. \qquad (4.2)$$

Definition 17. *Let P be a PSO that contains the social component $S(t)$ in the updating principle and ρ a disagreements apply rule. In order to obtain* ***particle swarm optimization with disagreements (PSOD)****, P_D, a "disagreement injector" is defined as follows:*

$$\Psi_{PSO}(P,\rho) = P_D. \qquad (4.3)$$

After the injection function Ψ_{PSO} is applied, the updating principle from eq. (4.1) in the new P_D is becoming:

$$\begin{aligned} X(t+1,x_i) &= \rho(\Delta_V,\beta_i,t) \\ &= \rho(\Delta_V, X(t,x_i) + V(t,i) + C(t,x_i,y_i) + S(t,x_i,\hat{y}_i) + \zeta, t) \\ &= X(t,x_i) + V(t,i) + C(t,x_i,y_i) + D_i(S(t,x_i,\hat{y}_i)) + \zeta, \\ & C(t,x_i,y_i) \to y_i, S(t,x_i,\hat{y}_i) \to D_i(\hat{y}_i), D_i \in \Delta_V, \forall i \in \overline{1,s}. \end{aligned} \qquad (4.4)$$

Disagreements are a special operator in PSO and the whole concept is designed to be applied only on the social component, which can have any particular implementation and which is detoured without interfering with the rest of components. For chosen individuals in an iteration, their social component will not point to neighborhood's best, $\hat{y}_i$, but towards a new point, $D_i(\hat{y}_i)$ around the neighborhood's best.

One of the most important points in the design philosophy for disagreements is not to alter too much the internals of the original algorithm and to keep things simple.

4.1.2. Implementations

There can be imagined many ways to implement a disagreement for PSO: particles can disagree as part of a learning process or a result of a complex social interaction, but unfortunately, complexity can only lead to a high computational cost.

In the research report [57], based on conclusions provided in [77] and in [102], this thesis' author advocated the increase of the local swarm the entropy in order to prevent swarm explosion and increase diversity. The most efficient way entropy can be increased is to use randomness injection. It seems like a cheap option to use randomness injection to simulate a disagreement because other approaches like automated learning or using extra memory and calculations can be more expensive. By doing so, this implementation variant for PSODs gain the advantages of *memetic particle swarm optimizers* - combinations of a PSO algorithm with a local search method, based on the concept of *memes* ([27]), as in [77] or [78] -, while retaining the low computational cost of the original PSO. There are a few added parameters and most of the time they do not need per-problem tuning.

By using pure randomness injection, the disagreements implementation provided in this thesis is following the "keep it simple" principle.

4.2. 6σ-PSOD Operator

4.2.1. Description

In order to get a glimpse on how disagreements act upon PSO, there was designed and tested a simple operator that imitates the real world proportion of disagreements and uses randomness injection across two neighborhood areas in search space to emulate it - the 6σ-PSOD operator. Its full presentation is taken from this thesis author's work, published in [58].

In its discovery phase, the 6σ-PSOD operator was shaped based on the assumption that real-world disagreements have a Gaussian distribution across a given population and that disagreements can be *partial* and *extreme*. Partial disagreements affect more members of a group than extreme disagreements do.

To make disagreements available in any PSO, an injector function is needed. Based on (4.3), the 6σ injector is defined as follows:

Definition 18. *Let P be a particle swarm optimization algorithm that has a social component. The function that injects in P a set of disagreements - $\Delta_{6\sigma}$, with an apply rule - $\rho_{6\sigma}$, and transforms it into a particle swarm optimization with disagreements following the 6σ rule, namely a 6σ-PSOD algorithm - $P_{D6\sigma}$, is defined as:*

$$\Psi_{6\sigma-PSOD}(P) = \Psi_{PSO}(P,\rho_{6\sigma}) = P_{D6\sigma}. \tag{4.5}$$

In this type of PSOD, the updating principle from eq. (4.4) becomes:

$$\begin{aligned} X(t+1,x_i) &= \rho_{6\sigma}(\Delta_{6\sigma},\beta_i,t) \\ &= \rho_{6\sigma}(\Delta_{6\sigma},X(t,x_i)+V(t,i)+C(t,x_i,y_i)+S(t,x_i,\hat{y}_i)+\zeta,t) \\ &= X(t,x_i)+V(t,i)+C(t,x_i,y_i)+D_{6\sigma i}(S(t,x_i,\hat{y}_i))+\zeta, \\ & C(t,x_i,y_i)\to y_i, S(t,x_i,\hat{y}_i)\to D_{6\sigma i}(\hat{y}_i), D_{6\sigma i}\in\Delta_{6\sigma}, \forall i\in\overline{1,s}. \end{aligned} \tag{4.6}$$

For $\Psi_{6\sigma-PSOD}$ the set of disagreements (the Δ_V from the relation (43) that contains the disagreements, $D_{6\sigma i}$) is composed of:

$$\Delta_{6\sigma} = \{\varnothing_D, D_p, D_e\}. \tag{4.7}$$

The first member in the set from (47) - $\varnothing_D$, is the *"no-op disagreement"*, meaning that no disagreement takes place. Being an identity function, when applied to the social component it yields $\varnothing_D(S) = S$. This non-disagreement is applied to most of the particles in iteration because the majority of the particles follow the mainstream, they do not disagree.

The first real disagreement employed here is D_p. It is a partial disagreement because it tempers and skews the social component around its very vicinity. This type of disagreement takes place quite often in a 6σ-PSOD algorithm and represents that part of the population that do not exactly follow the main trend but has related beliefs to the mainstream. In this PSOD, it is used to enhance the exploitation in the neighborhood of the current solutions. It is a function that multiplies member-wise (a Hadamard product) the social component S by a vector r, which has its components uniformly distributed in the interval $[-1,+1]$.

$$D_p(S) = r_p \otimes S, r_p \sim \mathrm{U}(-1,+1), p \in \overline{1,|r|}. \tag{4.8}$$

The second disagreement is the extreme disagreement D_e, called so because it implies an extreme amplification of the social component S. It represents the individuals that hold extreme opinions in society. In this PSOD, it is utilized to enhance the exploration beyond the current capabilities of the old PSOs. Mathematically, it is a Hadamard product between the social component and a vector r containing random uniformly distributed values in the intervals $[-2,-1]$ and $[+1,+2]$:

$$\begin{aligned} &D_e(S) = r_e \otimes S, r_e = r_p + sgn(r_p), \\ &r_p \sim U(\text{-}1,1)\ p \in \overline{1,|r|}. \end{aligned} \tag{4.9}$$

A rudimentary visualization of the two types of disagreements is given in Fig. 4.1. In concentric circles, two areas are shown: the inner area is where the potential result of D_p can end up for partial disagreements, while for extreme disagreements the outer area between circles is where can be the potential result of D_e. The end arrow for social component S will finally point into one of these two areas if disagreements are invoked according to the formula 4.13.

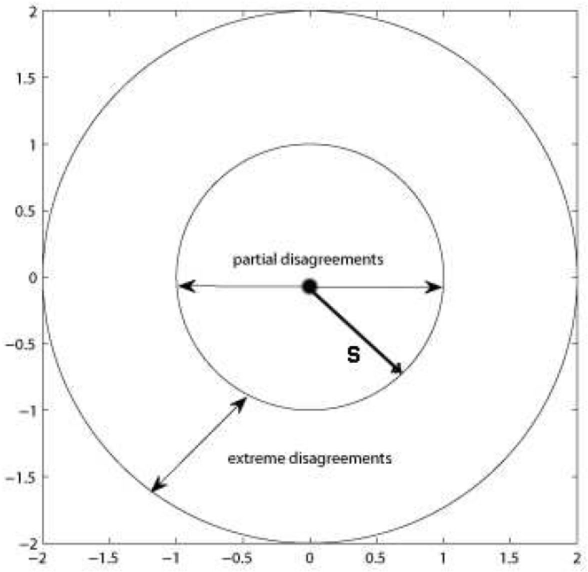


Figure 4.1: Distribution of disagreement types in concentric circles.

Before the run a Gaussian distribution $\theta = \mathrm{N}(\mu_{6\sigma}, \sigma_{6\sigma}^2)$ is considered as a reference. Under the 6σ rule, at each iteration t, for each particle i, a $\theta_1(t,i) \sim \mathrm{N}(\mu_{6\sigma}, \sigma_1^2)$ is generated such that $\sigma_{6\sigma} \geq \sigma_1$. θ_1 is a parameter that depends on the initially set Gaussian distribution and can be used in experiments to control how much disagreements are injected. In Fig. 4.2 it is shown how θ_1 acts as a filter.

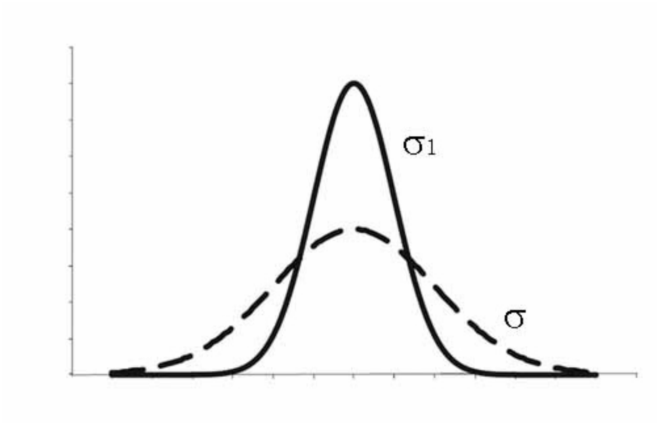


Figure 4.2: Filtering disagreements.

For θ, the following Gaussian regions are defined:

A no disagreements region: accounts for approx. *68.2%* of the bell curve (first two σ s) and it is defined as:

$$R_{1,2\sigma} = (\mu_{6\sigma} - \sigma_{6\sigma}) \cup (\mu_{6\sigma} + \sigma_{6\sigma}). \quad (4.10)$$

A partial disagreements region: accounts for approx. *27.2%* of the bell curve (next two σ s) and it is defined as:

$$R_{3,4\sigma} = (\mu_{6\sigma} - 2\sigma_{6\sigma}, \mu_{6\sigma} - \sigma_{6\sigma}] \cup [\mu_{6\sigma} + \sigma_{6\sigma}, \mu_{6\sigma} + 2\sigma_{6\sigma}). \quad (4.11)$$

An extreme disagreements region: accounts for approx. *4.6%* of the bell curve (next two σ s and the rest of what remains under the graphic of the Gaussian function) and it is defined as:

$$R_{5,6\sigma} = (-\infty, \mu_{6\sigma} - 2\sigma_{6\sigma}] \cup [\mu_{6\sigma} + 2\sigma_{6\sigma}), +\infty). \quad (4.12)$$

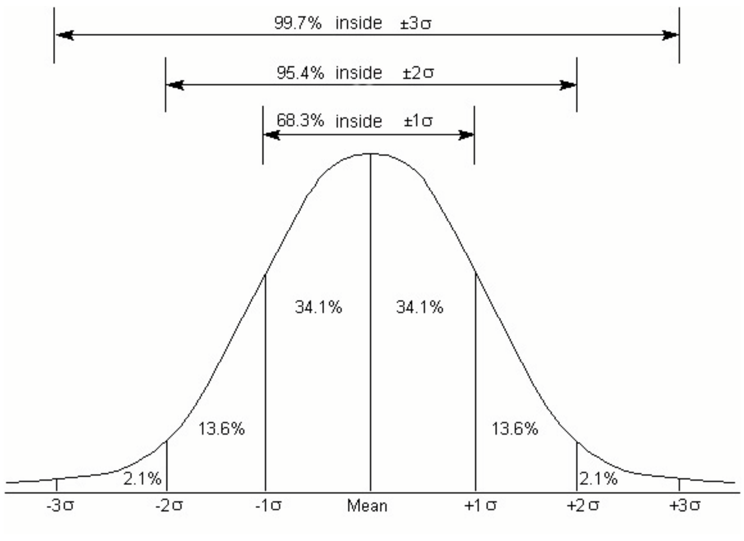


Figure 4.3: 6σ regions.

Fig. 4.3 illustrates the 6-σ regions for θ. Based on the above provided equations, the apply rule (the disagreements selector function) is defined as follows:

$$\rho_{6\sigma}(\Delta_{6\sigma},\beta_i,t)=\begin{cases}\varnothing_D(S), & \text{if } \theta_1(t,i)\in R_{1,2\sigma}\\ D_p(S), & \text{if } \theta_1(t,i)\in R_{3,4\sigma}\\ D_e(S), & \text{if } \theta_1(t,i)\in R_{5,6\sigma}\end{cases} \quad . \tag{4.13}$$

The 6σ-PSOD operator - $\rho_{6\sigma}$, makes sure that there is a majority of particles not affected by disagreements, that there is a minority of individuals that partially agrees and a small minority that totally disagrees.

4.2.2. Experimental Results

Setup

In order to determine the usefulness and the performance of the 6σ-PSOD operator, **race-testing methodology** was used as explained in Subsection 3.3. In the initial **discovery phase** the new concept applied to SPSO yielded mixed results on two randomly picked test configurations, but in subsequent experiments it was **confirmed** that for most benchmarks a filtering value of $\sigma_1 = 0.7$ provides superior performance.

A short performance overview for how disagreements act upon standard PSO (SPSO) and the à la Pedersen simplified PSO (PSO-VG) for all 10 benchmarks from Subsection 4.3 is given below. The two corresponding disagreements-enabled algorithms are $SPSOD_{6\sigma}$ and $PSO-VGD_{6\sigma}$. Considered cases cover swarms with 25 and 50 particles and benchmarks problems in a high dimensional environment - 30 and 50 dimensions, a configuration which surrogates most possible nowadays real-world problems. The mean best fitness, its standard deviation, the median and how many partial disagreements per iteration (p.d.i.) and extreme disagreements per iteration (e.d.i.) were involved are provided as output values from experiments that averaged the performance from 100 runs.

The benchmarks are classified under several categories: plateau functions, shifted multi-modal functions, low dimensional problems, regular multi-modal functions and other types of problems.

A Plateau Function

Generalized Rosenbrock (LF_1) is a tricky plateau function, unimodal and non-separable that represents well the category of plateau functions. Finding better solutions on LF_1 prophesies a good approach. Results are provided in Table 4.1.

Table 4.1: 6σ -PSOD results for LF_1 .

algorithm	d	s	mean	std. dev.	median	p.d.i.	e.d.i.
SPSO	30	25	7145.40	9083.21	3859.55	0.00	0.00
SPSOD$_{6\sigma}$	30	25	1392.72	3714.24	377.50	4.02	0.12
PSO–VG	30	25	127751.53	227428.89	32393.18	0.00	0.00
PSO–VGD$_{6\sigma}$	30	25	37972.12	98239.48	10531.20	3.31	0.07
SPSO	30	50	462.57	1677.74	81.32	0.00	0.00
SPSOD$_{6\sigma}$	30	50	59.73	39.78	65.64	6.93	0.16
PSO–VG	30	50	4909.38	13650.26	896.37	0.00	0.00
PSO–VGD$_{6\sigma}$	30	50	1415.92	3361.93	300.05	6.74	0.24
SPSO	50	25	65604.93	110255.89	35342.06	0.00	0.00
SPSOD$_{6\sigma}$	50	25	25250.61	94110.65	4315.54	3.32	0.09
PSO–VG	50	25	658684.63	466908.89	521877.29	0.00	0.00
PSO–VGD$_{6\sigma}$	50	25	377445.38	426439.43	185804.74	3.35	0.09
SPSO	50	50	9708.33	41110.93	1788.16	0.00	0.00
SPSOD$_{6\sigma}$	50	50	549.90	962.62	317.56	6.29	0.17
PSO–VG	50	50	140868.25	222987.33	66829.56	0.00	0.00
PSO–VGD$_{6\sigma}$	50	50	102323.20	212938.27	31973.96	7.14	0.21

For 30 dimensions, it can be easily noticed that for $SPSOD_{6\sigma}$ vs. SPSO with 25 particles in swarm, the mean best fitness across runs is 5.13 times better, while for the social only variants $PSO-VGD_{6\sigma}$ is 3.36 times better. This configuration with 25 particles in 30 dimensions is of higher interest than the one with 50 particles in the same dimensions, where the 6σ approach scored better also (7.74 and 3.46 times better, respectively) for the obvious reason that it implies a smaller number of particles and a lower computational cost. A table with the Euclidean distance between means for the case with 25 particles and 30 dimensions in LF_1 is provided (see Table 4.2). Fig. 4.4 shows the magnitude of the improvement.

Table 4.2: Euclidean distance between means (LF_1 in 30 dimensions, 25 particles).

algorithm	**SPSO**	**SPSOD$_{6\sigma}$**	**PSO–VG**	**PSO–VGD$_{6\sigma}$**
SPSO	0.00	5752.68	120606.12	30826.71
SPSOD$_{6\sigma}$	**5752.68**	0.00	126358.81	36579.40
PSO–VG	120606.12	126358.81	0.00	89779.40
PSO–VGD$_{6\sigma}$	30826.71	36579.40	**89779.40**	0.00

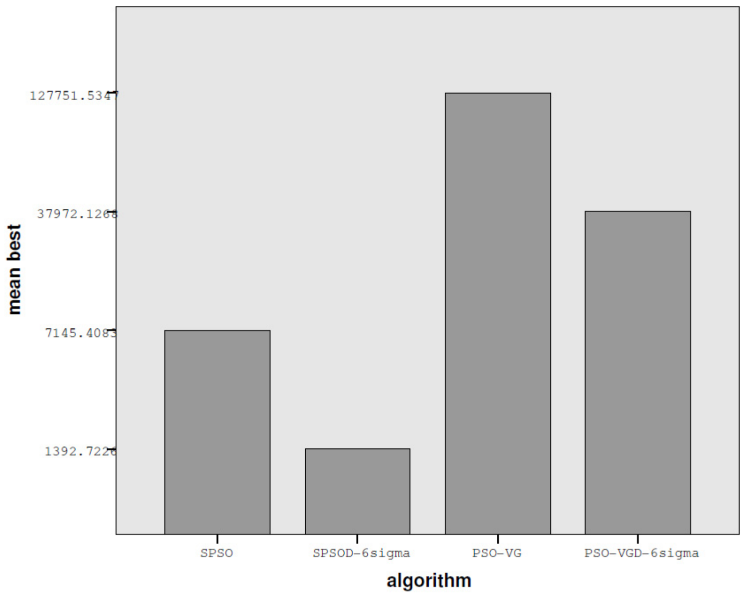


Figure 4.4: Overview of 6σ -PSOD improvements with 25 particles for LF_1 in 30 dimensions.

The graphs with the evolution of the best individual from Fig. 4.5 and Fig. 4.6 show how the above results are possible. It is amazing that the injection of disagreements does not disturb $PSO - VGD_{6\sigma}$ making it run off the rails, but it actually helps it find a better path in the search space.

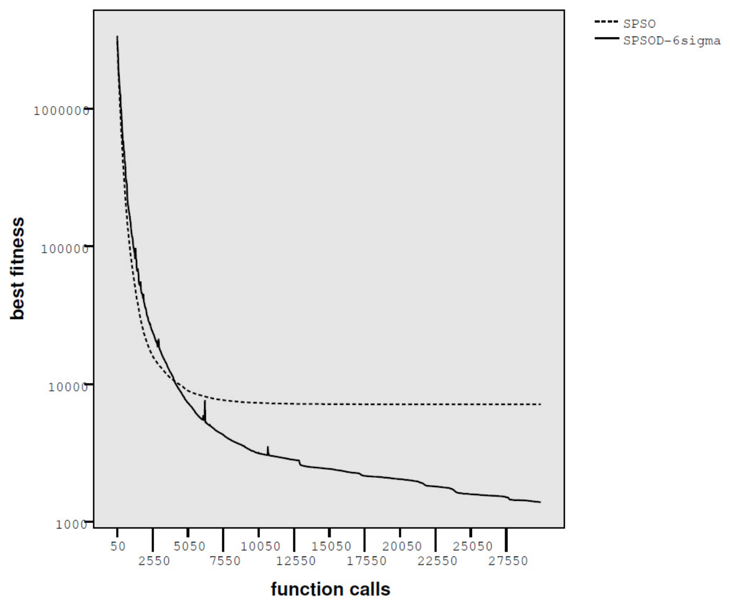


Figure 4.5: $SPSOD_{6\sigma}$ vs. SPSO best fitness graph with 25 particles for LF_1 in 30 dimensions.

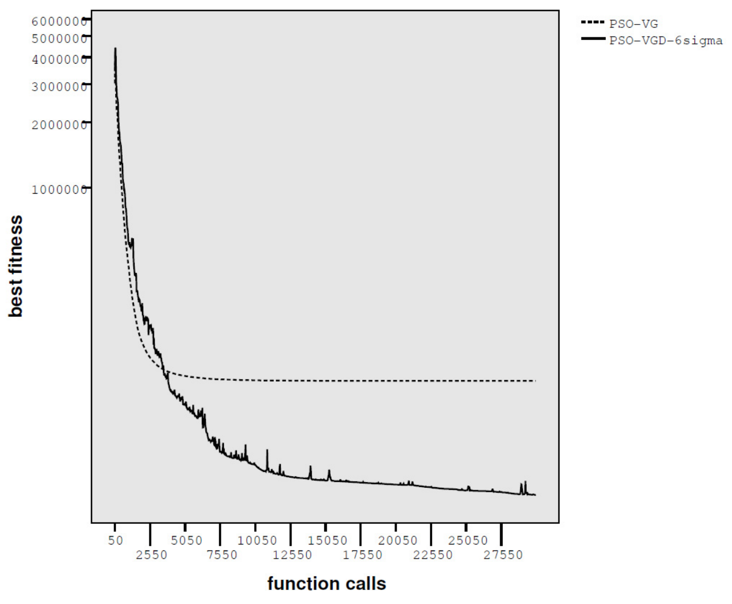


Figure 4.6: $PSO-VGD_{6\sigma}$ vs. PSO-VG best fitness graph with 25 particles for LF_1 in 30 dimensions.

The secret ingredient for this huge improvement lies in the enhanced population diversity, as presented in Fig. 4.7 and Fig. 4.8.

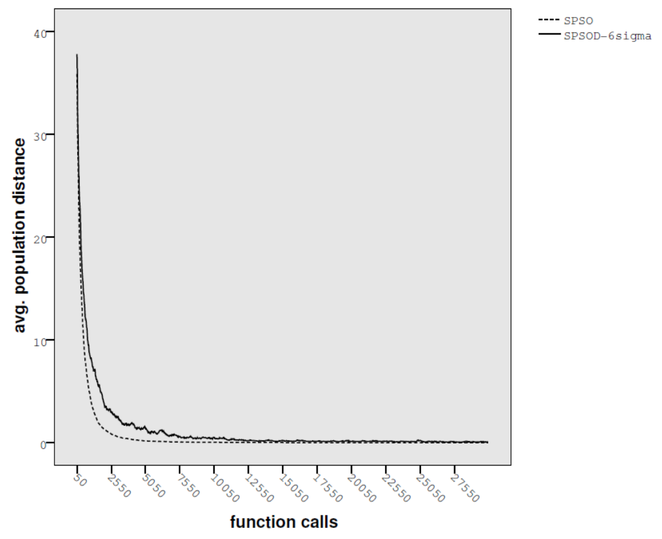


Figure 4.7: $SPSOD_{6\sigma}$ vs. SPSO avg. population distance graph with 25 particles for LF_1 in 30 dimensions.

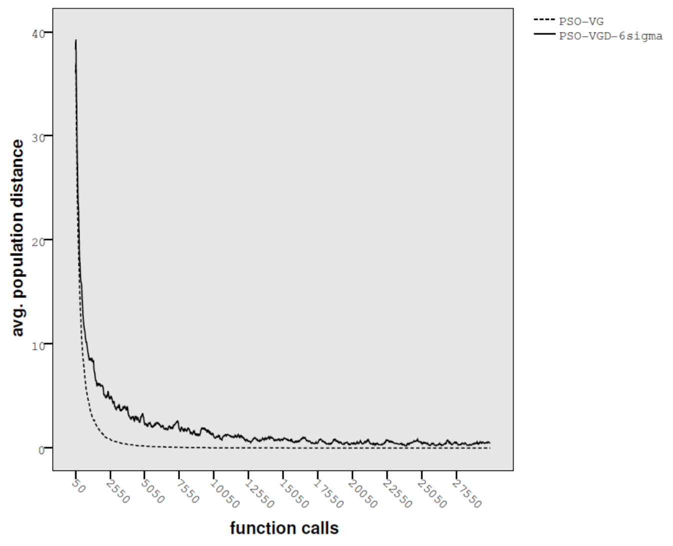


Figure 4.8: $PSO-VGD_{6\sigma}$ vs. PSO-VG avg. population distance graph with 25 particles for LF_1 in 30 dimensions.

The worst individual moves swiftly in the case of 6σ variants, thus being most of the time better than its original counterpart (see Fig. 4.9 and Fig. 4.10), which stagnates.

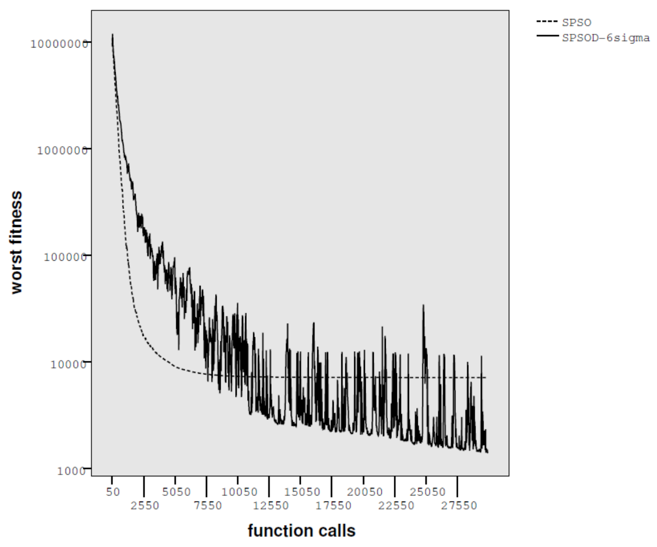


Figure 4.9: $SPSOD_{6\sigma}$ vs. SPSO worst fitness graph with 25 particles for LF_1 in 30 dimensions.

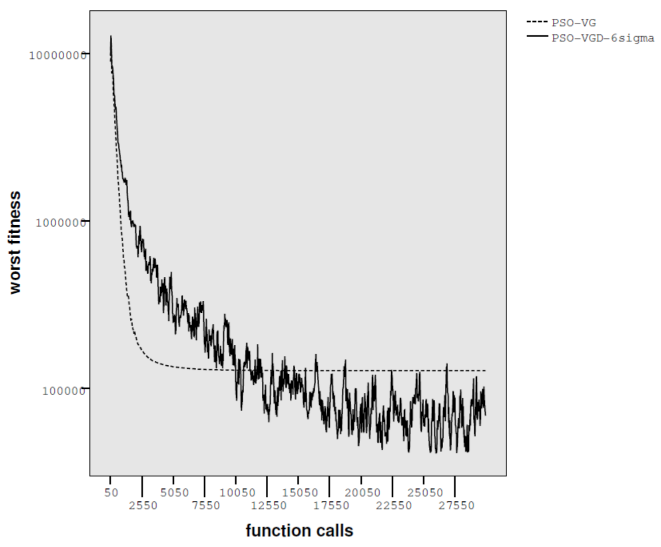


Figure 4.10: $PSO-VGD_{6\sigma}$ vs. PSO-VG worst fitness graph with 25 particles for LF_1 in 30 dimensions.

Results for 50 dimensions from Table 4.1 show a 2.59 and 1.74 times improvement in the case of PSODs for 25 particles in swarm and 17.65 and 1.37 times improvement for 50 particles.

For LF_1, the results using the 6σ-PSOD operator (with $\sigma_1 = 0.7$ - which is limiting the number of disagreements) are far better than using the original algorithms.

Shifted Multi-Modal Functions

For Shifted Sphere (LF_2), a quite easy optimization test, Table 4.3 shows without any doubt a dramatic improvement when using disagreements.

Table 4.3: 6σ-PSOD results for LF_2.

algorithm	d	s	mean	std. dev.	median	p.d.i.	e.d.i.
SPSO	30	25	2464.79	2510.30	1713.96	0.00	0.00
$\text{SPSOD}_{6\sigma}$	30	25	609.54	1396.12	103.37	3.27	0.11
PSO–VG	30	25	7222.44	5196.16	5863.67	0.00	0.00
$\text{PSO–VGD}_{6\sigma}$	30	25	4385.17	4825.14	2678.87	3.74	0.10
SPSO	30	50	65.65	268.94	0.12	0.00	0.00
$\text{SPSOD}_{6\sigma}$	30	50	19.17	106.52	0.01	6.79	0.18
PSO–VG	30	50	1064.66	1466.07	459.62	0.00	0.00
$\text{PSO–VGD}_{6\sigma}$	30	50	761.98	1075.07	443.85	7.05	0.23
SPSO	50	25	8592.50	5920.26	7049.39	0.00	0.00
$\text{SPSOD}_{6\sigma}$	50	25	3460.38	4588.08	1687.94	3.53	0.10
PSO–VG	50	25	25868.12	10548.08	24726.98	0.00	0.00
$\text{PSO–VGD}_{6\sigma}$	50	25	14677.50	11569.13	11528.80	3.40	0.09
SPSO	50	50	1419.61	2085.67	645.23	0.00	0.00
$\text{SPSOD}_{6\sigma}$	50	50	461.51	853.36	202.65	7.23	0.22
PSO–VG	50	50	7441.48	5650.67	5617.75	0.00	0.00
$\text{PSO–VGD}_{6\sigma}$	50	50	7336.57	5227.04	5811.31	6.59	0.17

LF_3, CEC 2005's Shifted Rosenbrock, is the shifted variant of LF_1, whose results were already presented above. The big positive performance gap that is brought by disagreements can be studied in Table 4.4. The very good results show that the 6σ approach can also work in disturbed environments and on test functions that are not prone to biases. The only negative case here is the case of PSO-VG with 25 particles in 30 dimensions, where the mean best fitness is slightly higher for the 6σ-PSOD variant, but it can be noticed that the median was better for the 6σ. The other cases are clearly in favor of disagreements. By testing the original and the shifted variants of Rosenbrock function with good results, it can be concluded that the new approach with disagreements should be definitively used on plateau functions.

Table 4.4: 6σ -PSOD results for LF_3 .

algorithm	d	s	mean	std. dev.	median	pdi; edi
SPSO	30	25	210547557	405846769	43327495	0.00 ; 0.00
SPSOD$_{6\sigma}$	30	25	5995260	19742998	201936	3.62 ; 0.09
PSO–VG	30	25	913750237	1270657873	381281080	0.00 ; 0.00
PSO–VGD$_{6\sigma}$	30	25	751270527	4656362012	87525797	3.39 ; 0.09
SPSO	30	50	3241751	14341316	2159	0.00 ; 0.00
SPSOD$_{6\sigma}$	30	50	18565	130904	243	7.29 ; 0.21
PSO–VG	30	50	76838896	157427839	24563312	0.00 ; 0.00
PSO–VGD$_{6\sigma}$	30	50	92001687	317979680	9643907	7.26 ; 0.25
SPSO	50	25	1278577946	2003933202	546570301	0.00 ; 0.00
SPSOD$_{6\sigma}$	50	25	794216787	5090544115	63581155	3.44 ; 0.08
PSO–VG	50	25	8996883580	9259990476	6211883021	0.00 ; 0.00
PSO–VGD$_{6\sigma}$	50	25	4064572152	5691695556	1575426795	3.65 ; 0.09
SPSO	50	50	42713105	86182326	17330117	0.00 ; 0.00
SPSOD$_{6\sigma}$	50	50	8280624	28173160	370156	6.99 ; 0.21
PSO–VG	50	50	1592822113	1901558466	777238443	0.00 ; 0.00
PSO–VGD$_{6\sigma}$	50	50	1031064717	1544552416	708645978	7.00 ; 0.22

Shifted Rastrigin (LF_4) is a test function which resembles a large basin and has many local optima. Table 4.5 provides evidence on how the disagreements-enabled particle swarm optimizers perform better on this test function too. The improvement is not as spectacular as in the previous cases, this time it can only be measured in percents, but it is still worthwhile in the context of optimization, where every gained decimal matters. The difficult case with 25 particles in swarm and in 50 dimensions is considered for analysis: the evolution of the best fitness graph is the one provided in Fig. 4.11, the evolution of the worst fitness found by the swarm is presented in 4.12 and the population diversity is shown in 4.13. In contrast with the graphs for LF_1, the graphs for LF_4 show a smooth evolution for the 6σ -PSOD algorithms also, which finally manage to find better solutions than their original counterparts do.

Table 4.5: 6σ -PSOD results for LF_4 .

algorithm	d	s	mean	std. dev.	median	p.d.i.	e.d.i.
SPSO	30	25	141.70	37.73	139.14	0.00	0.00
SPSOD$_{6\sigma}$	30	25	112.40	33.99	110.77	3.46	0.08
PSO–VG	30	25	223.58	47.31	218.86	0.00	0.00
PSO–VGD$_{6\sigma}$	30	25	179.66	50.53	173.43	3.41	0.11
SPSO	30	50	77.05	21.75	72.54	0.00	0.00
SPSOD$_{6\sigma}$	30	50	68.20	18.36	69.21	7.03	0.15
PSO–VG	30	50	132.98	32.78	127.52	0.00	0.00
PSO–VGD$_{6\sigma}$	30	50	124.87	34.48	119.60	6.58	0.18
SPSO	50	25	309.46	60.31	313.20	0.00	0.00
SPSOD$_{6\sigma}$	50	25	261.43	63.91	254.22	3.62	0.09
PSO–VG	50	25	493.50	86.74	487.96	0.00	0.00
PSO–VGD$_{6\sigma}$	50	25	430.12	78.53	435.22	3.73	0.10
SPSO	50	50	175.38	41.06	171.86	0.00	0.00
SPSOD$_{6\sigma}$	50	50	154.35	33.79	149.69	7.12	0.21
PSO–VG	50	50	313.71	62.58	312.94	0.00	0.00
PSO–VGD$_{6\sigma}$	50	50	292.09	58.70	293.06	7.00	0.22

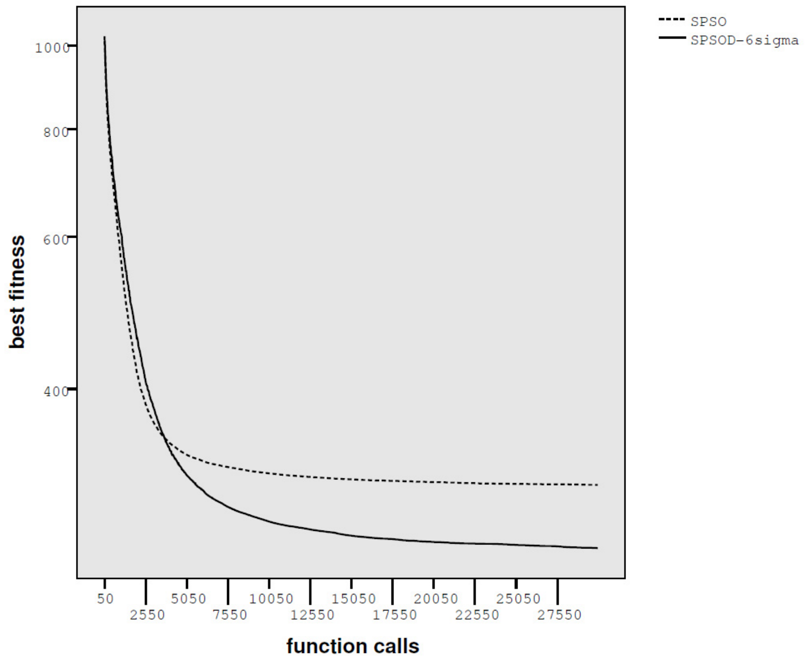


Figure 4.11: $SPSOD_{6\sigma}$ vs. SPSO best fitness graph for LF_4 with 25 particles in 50 dimensions.

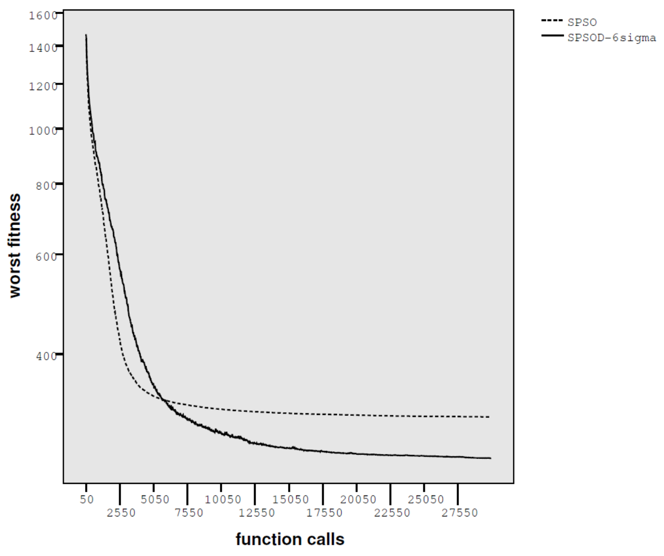


Figure 4.12: $SPSOD_{6\sigma}$ vs. SPSO worst fitness graph for LF_4 with 25 particles in 50 dimensions.

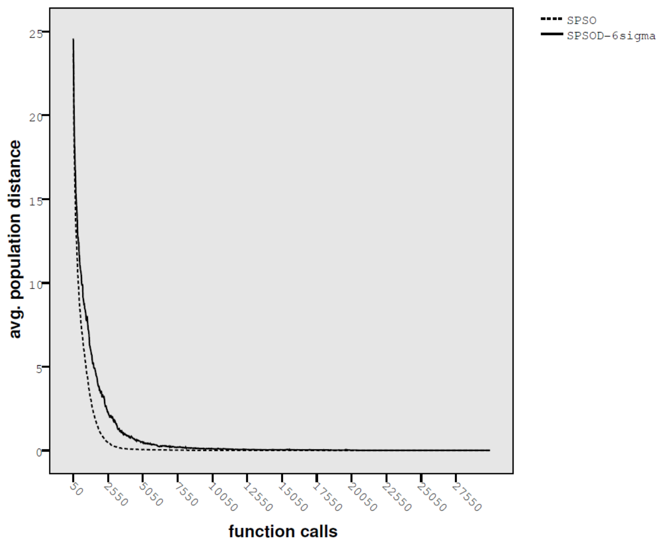


Figure 4.13: $SPSOD_{6\sigma}$ vs. SPSO avg. population distance graph for LF_4 with 25 particles in 50 dimensions.

Shifted Schwefel 1.2 (LF_5) is another hard problem to test. Results from Table 4.6 show that the only case when the original algorithms perform better than the *6σ* enhanced ones is for swarms with 50 particles in 50 dimensions. However, the performance difference is small, meaning that it would be better to apply disagreements anyhow because an eventual penalty is insignificant and the benefit in the other cases is high.

Table 4.6: 6σ -PSOD results for LF_5 .

algorithm	d	s	mean	std. dev.	median	p.d.i.	e.d.i.
SPSO	30	25	9623.30	6300.01	8032.85	0.00	0.00
SPSOD$_{6\sigma}$	30	25	5538.67	4590.39	4359.27	3.05	0.09
PSO–VG	30	25	33695.51	13189.61	31326.23	0.00	0.00
PSO–VGD$_{6\sigma}$	30	25	24058.02	12484.09	22144.86	3.51	0.10
SPSO	30	50	2754.11	2993.46	1720.71	0.00	0.00
SPSOD$_{6\sigma}$	30	50	1645.28	2092.90	895.94	7.05	0.19
PSO–VG	30	50	12518.16	8561.35	10399.03	0.00	0.00
PSO–VGD$_{6\sigma}$	30	50	10244.92	6676.21	8913.05	6.77	0.23
SPSO	50	25	39689.11	13928.32	39157.67	0.00	0.00
SPSOD$_{6\sigma}$	50	25	27766.61	11003.62	25912.75	3.65	0.09
PSO–VG	50	25	93174.26	25414.81	87288.75	0.00	0.00
PSO–VGD$_{6\sigma}$	50	25	75361.03	21747.51	72555.14	3.24	0.08
SPSO	50	50	29820.25	9891.27	29721.57	0.00	0.00
SPSOD$_{6\sigma}$	50	50	31527.85	11403.90	28760.42	6.26	0.17
PSO–VG	50	50	56893.48	19790.34	55503.57	0.00	0.00
PSO–VGD$_{6\sigma}$	50	50	58695.99	17100.10	56977.46	7.12	0.20

A Low Dimensional Problem

Himmelblau (LF_6) is a low dimensional plateau test function. Because all algorithms found easily the solution, having the same performance, it is not recommended to use disagreements for solving this problem. The original algorithms already solve it successfully. What is important to notice is that using the 6σ model of disagreements did not impede the performance of the original algorithm. Table 4.7 shows the results just for the record.

Table 4.7: 6σ -PSOD results for LF_6 .

algorithm	s	mean	std. dev.	median	p.d.i.	e.d.i.
SPSO	25	0.00	0.00	0.00	0.00	0.00
SPSOD$_{6\sigma}$	25	0.00	0.00	0.00	3.45	0.08
PSO–VG	25	0.00	0.00	0.00	0.00	0.00
PSO–VGD$_{6\sigma}$	25	0.00	0.00	0.00	2.78	0.04
SPSO	50	0.00	0.00	0.00	0.00	0.00
SPSOD$_{6\sigma}$	50	0.00	0.00	0.00	6.26	0.18
PSO–VG	50	0.00	0.00	0.00	0.00	0.00
PSO–VGD$_{6\sigma}$	50	0.00	0.00	0.00	7.11	0.25

Regular Multi-Modal Functions

Griewank, LF_7, is one of the functions on which the behavior of algorithms on highly multi-modal functions can be studied. Results from Table 4.8 are self-explanatory: overall, there is a significant improvement when using disagreements.

Table 4.8: 6σ -PSOD results for LF_7 .

algorithm	d	s	mean	std. dev.	median	p.d.i.	e.d.i.
SPSO	30	25	9.91	14.72	4.33	0.00	0.00
SPSOD$_{6\sigma}$	30	25	3.30	9.64	1.05	3.21	0.11
PSO–VG	30	25	66.03	52.34	49.87	0.00	0.00
PSO–VGD$_{6\sigma}$	30	25	37.28	45.41	18.74	3.64	0.11
SPSO	30	50	1.42	2.70	0.42	0.00	0.00
SPSOD$_{6\sigma}$	30	50	0.16	0.76	0.03	7.56	0.20
PSO–VG	30	50	12.44	17.91	5.99	0.00	0.00
PSO–VGD$_{6\sigma}$	30	50	8.40	19.23	1.88	7.06	0.18
SPSO	50	25	45.48	35.00	34.59	0.00	0.00
SPSOD$_{6\sigma}$	50	25	13.43	21.67	4.71	3.36	0.08
PSO–VG	50	25	245.70	96.28	240.27	0.00	0.00
PSO–VGD$_{6\sigma}$	50	25	166.51	84.61	156.08	3.32	0.08
SPSO	50	50	12.52	21.06	5.24	0.00	0.00
SPSOD$_{6\sigma}$	50	50	5.01	11.52	2.15	7.28	0.19
PSO–VG	50	50	74.59	52.11	59.00	0.00	0.00
PSO–VGD$_{6\sigma}$	50	50	73.30	65.37	58.25	6.64	0.18

Ackley, LF_8, is a function that has the global optimum inside a long funnel. Nevertheless, the tight road to the best point in the search space is paved with many local optima in which many optimizers fall. This is another important benchmark function in which the 6σ -PSOD shows its superiority, as it scores better in all tests.

Table 4.9: 6σ -PSOD results for LF_8 .

algorithm	d	s	mean	std. dev.	median	p.d.i.	e.d.i.
SPSO	30	25	11.54	2.86	11.75	0.00	0.00
$\text{SPSOD}_{6\sigma}$	30	25	9.88	2.97	9.38	3.32	0.08
PSO–VG	30	25	17.68	2.13	18.55	0.00	0.00
$\text{PSO–VGD}_{6\sigma}$	30	25	16.84	2.52	17.52	3.50	0.09
SPSO	30	50	4.12	2.49	3.98	0.00	0.00
$\text{SPSOD}_{6\sigma}$	30	50	1.57	1.30	1.34	7.17	0.20
PSO–VG	30	50	11.01	3.77	10.47	0.00	0.00
$\text{PSO–VGD}_{6\sigma}$	30	50	9.41	4.33	8.49	6.82	0.18
SPSO	50	25	15.10	2.02	15.36	0.00	0.00
$\text{SPSOD}_{6\sigma}$	50	25	13.49	2.84	14.14	3.60	0.10
PSO–VG	50	25	19.31	0.47	19.42	0.00	0.00
$\text{PSO–VGD}_{6\sigma}$	50	25	19.19	0.41	19.21	3.34	0.07
SPSO	50	50	8.22	2.77	7.74	0.00	0.00
$\text{SPSOD}_{6\sigma}$	50	50	4.74	2.56	3.91	7.00	0.23
PSO–VG	50	50	16.60	2.12	17.13	0.00	0.00
$\text{PSO–VGD}_{6\sigma}$	50	50	16.53	2.95	17.41	7.12	0.20

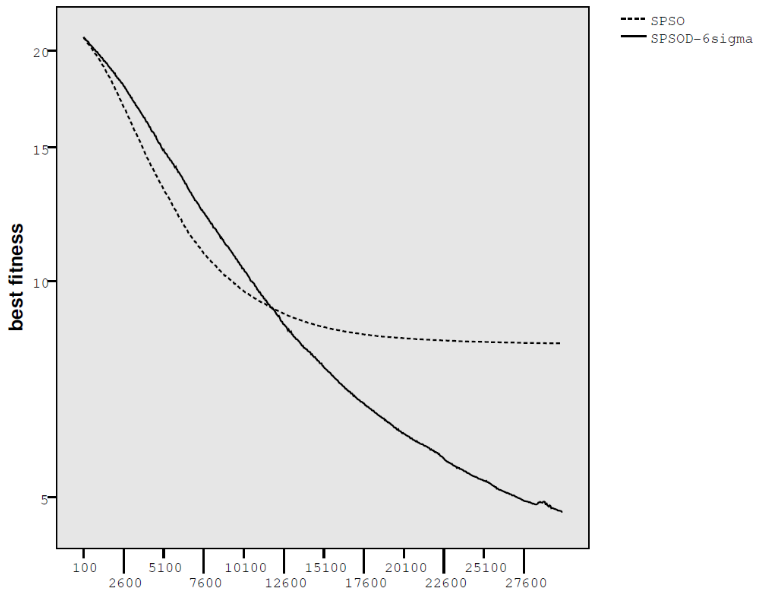


Figure 4.14: $SPSOD_{6\sigma}$ vs. SPSO best fitness graph for LF_8 with 50 particles in 50 dimensions.

Best fitness evolution graphs are provided in Fig. 4.14 for SPSO - in which $SPSOD_{6\sigma}$ clearly wins, and in Fig. 4.15 - in which $PSO-VGD_{6\sigma}$ is slightly better.

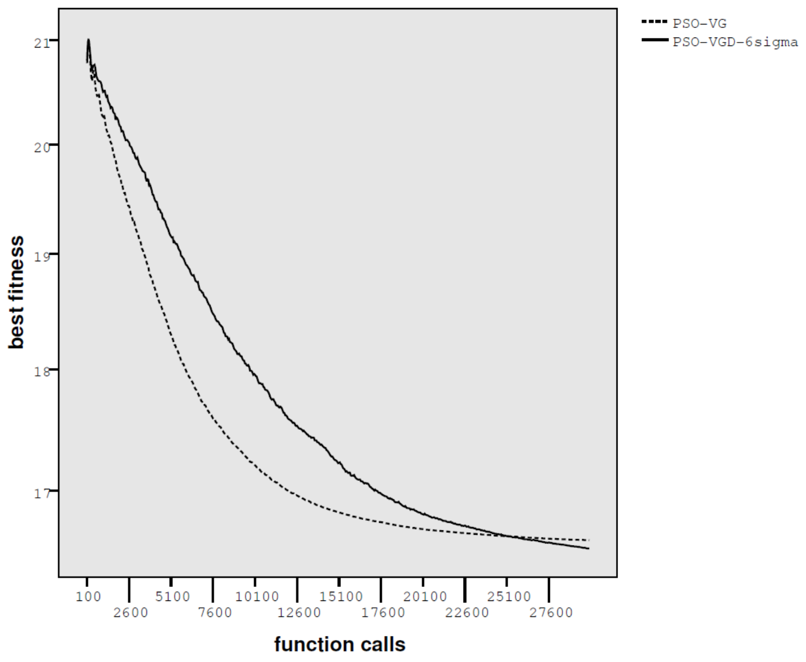


Figure 4.15: $PSO-VGD_{6\sigma}$ vs. PSO-VG best fitness graph for LF_8 with 50 particles in 50 dimensions.

The ability to find better results in a tight tunnel like Ackley's function proves that 6σ-PSODs have a better local focus. This is obtained using partial disagreements that are refining already good solutions, providing an enhanced exploitation.

Other Functions

The last functions in test are Bohachevsky 1 (LF_9) and Kursawe (LF_{10}). The results for both are once again better for the proposed new approach and do not need extra comments. Tables 4.10 and 4.11 are given for reference.

The only exceptional case for LF_9, "exceptional" meaning that the original algorithm performed better, is PSO-VG with 50 particles in 50 dimensions, but compared with the rest of the results it can be considered a small penalty.

For Kursawe functions (better values are the ones closer to -10, the sum of both Kursawe global optima), there is only one exceptional case for PSO-VG in 50 dimensions, with 25 particles. All other cases are favorable to PSODs. Results obtained on Kursawe prove that improvements are not limited to single function environments, but they can come into effect to multi-objective optimization also.

Table 4.10: 6σ -PSOD results for LF_9 .

algorithm	d	s	mean	std. dev.	median	p.d.i.	e.d.i.
SPSO	30	25	48.82	43.94	32.25	0.00	0.00
SPSOD$_{6\sigma}$	30	25	16.89	27.41	10.50	3.57	0.10
PSO–VG	30	25	240.69	166.11	206.76	0.00	0.00
PSO–VGD$_{6\sigma}$	30	25	120.13	117.00	85.23	3.61	0.10
SPSO	30	50	10.26	14.93	6.29	0.00	0.00
SPSOD$_{6\sigma}$	30	50	3.99	2.56	3.56	6.65	0.22
PSO–VG	30	50	71.01	86.90	35.03	0.00	0.00
PSO–VGD$_{6\sigma}$	30	50	46.44	74.41	18.59	6.93	0.23
SPSO	50	25	191.90	138.81	151.31	0.00	0.00
SPSOD$_{6\sigma}$	50	25	89.98	95.12	55.90	3.55	0.09
PSO–VG	50	25	914.83	323.38	931.56	0.00	0.00
PSO–VGD$_{6\sigma}$	50	25	649.54	288.27	625.11	3.63	0.11
SPSO	50	50	82.04	78.86	60.64	0.00	0.00
SPSOD$_{6\sigma}$	50	50	33.58	24.35	28.48	7.11	0.22
PSO–VG	50	50	320.20	193.11	294.53	0.00	0.00
PSO–VGD$_{6\sigma}$	50	50	322.73	220.96	319.19	6.80	0.19

Table 4.11: 6σ -PSOD results for LF_{10} .

algorithm	d	s	mean	std. dev.	median	p.d.i.	e.d.i.
SPSO	30	25	-215.56	45.99	-235.89	0.00	0.00
SPSOD$_{6\sigma}$	30	25	-210.13	68.22	-249.15	3.50	0.09
PSO–VG	30	25	-208.42	21.77	-199.78	0.00	0.00
PSO–VGD$_{6\sigma}$	30	25	-199.74	35.96	-152.42	3.36	0.09
SPSO	30	50	-131.37	46.21	-144.75	0.00	0.00
SPSOD$_{6\sigma}$	30	50	-63.32	35.63	-46.79	7.07	0.20
PSO–VG	30	50	-168.82	26.90	-157.10	0.00	0.00
PSO–VGD$_{6\sigma}$	30	50	-123.58	47.12	-126.54	7.28	0.21
SPSO	50	25	-171.38	55.53	-189.75	0.00	0.00
SPSOD$_{6\sigma}$	50	25	-86.60	41.08	-52.12	3.68	0.10
PSO–VG	50	25	-128.21	32.85	-139.68	0.00	0.00
PSO–VGD$_{6\sigma}$	50	25	-164.11	45.16	-153.29	3.69	0.10
SPSO	50	50	-68.24	19.34	-64.98	0.00	0.00
SPSOD$_{6\sigma}$	50	50	-45.53	14.72	-30.57	7.04	0.20
PSO–VG	50	50	-135.53	24.49	-152.49	0.00	0.00
PSO–VGD$_{6\sigma}$	50	50	-90.53	25.54	-84.08	7.14	0.21

Convergence Rates

Finally, in order to see some target hits, the convergence accuracy was changed from 0.01 to $\varepsilon = 0.1$ and the termination criterion was set from 30000 to 100000 fitness evaluations. The results from Table 4.12 show the convergence rates for a shifted and a non-shifted multimodal, LF_5 and LF_7, respectively.

Table 4.12: Some convergence rates for 6σ -PSODs.

algorithm	LF_5	LF_7
SPSO	0.00	0.28
$SPSOD_{6\sigma}$	0.14	0.81
PSO–VG	0.0	0.0
$PSO\text{–}VGD_{6\sigma}$	0.00	0.14

With only one exception, $PSO-VGD_{6\sigma}$ solving LF_5, where the convergence rate is zero and it is the same as in the case of the original algorithm, the rest of the cases confirm the exceptional results of the disagreements-enabled algorithms from the previous tables in this chapter.

4.2.3. Concluding Remarks

Algorithms that are enhanced with 6σ -PSOD capabilities perform far better than their original versions. The performance improvement in most of the tests is significant. Results from all tests from all the functions and some best individual's fitness graphs for a particularly tuned filter $\sigma_1 = 0.7$ were shown to show that the new approach is superior across the whole test palette.

Through graphs showing the evolution of the average distance between individuals, it was shown that the 6σ approach is increasing the population diversity and entropy. A direct consequence is that even the worst individual is far improved compared to the worst individual from the original algorithm. Of course, the "best fitness" of the run is a few times better in the case of the tested PSODs. This advantage was gained through the use of the partial disagreements (approx. 3 per iteration for 25 particles in the swarm and around 6-7 for 25 particles in swarm) that enhance the local exploitation in a very simple way (e.g. simpler than memetic PSOs) and through the use of the extreme disagreements (approx. 0.1 and 0.2 per iteration for 25 and 50 particles in swarm, respectively) that enhance the exploitation.

This simple implementation of disagreements consisting in a 2-layer neighborhood structure proved that the concept of disagreements is feasible for PSO and there is a high probability that it could be feasible for related algorithms in swarm intelligence. There were cases in which disagreements injection caused high tides in the swarm and cases in which the evolution was smooth, but in all situations, using disagreements provided better results. The extra-added operations consist in presetting a reference θ, generating each generation a θ_1 for filtering and applying the disagreements rule. σ_1 makes the algorithms to work well for a value of *0.7* and it should not be tuned every time, however even if it is a new parameter it is also a new degree of freedom that can be used to the optimizer's advantage. For most computer systems, nowadays this means calling the same random routines used for other evolutionary operations and using an extra loop to apply disagreements. That means that the added computational cost is negligible.

The fact that most runs were successful on the social-only algorithm, PSO-VG, indicates that this approach can be used to alter the whole output of the update principle. This observation is of high interest for situations in which an evolutionary algorithm does not have a split, classical PSO-like updating principle, and that is the case of genetic algorithms. Indeed, it will be shown in Chapter 5 that the 6σ approach works brilliantly for real-valued genetic algorithms also.

4.3. Stagnation Management With Disagreements

4.3.1. Stagnation

Stagnation, a situation in which there is no improvement of the current solution for an amount of iterations, can be mitigated if when it is detected disagreements are arisen inbetween members of the swarm. This procedure resembles the real-life situations in which in times of crisis people start riots or revolutions.

Swarm stagnation management is an area where disagreements can be used to save the general outcome of the run, as shown by the author in [60].

Swarm stagnation appears when the following conditions are met (quote from [46]):

The particle swarm system is thought to be in stagnation, if arbitrary particle i 's history best position P_i and the total swarm's history best position P_g keep constant over some time steps.

In order to detect the swarm stagnation in PSO, [101] introduces the concept of "improvement ratio" taking into consideration also the velocities:

$$R = \left| \frac{1 - f_c / f_p}{1 - v_c / v_p} \right|, \tag{4.14}$$

where f_c is the current fitness value of the best particle, while f_p is the previous fitness and v_c is the current average velocity of all particles, while v_p is the previously recorded one. According to [101], when R is droping under an *a priori* value ε then the swarm enters into a stagnation period.

As pointed out by the author in [60], "the approach from [101] assumes that when stagnation occurs the velocities tend to 0. However, this is valid only in situations when particles get trapped into local minima, while for very densely spiked or plateau functions this could not be true." Therefore, a more relaxed criterion is needed to establish when stagnation appears that covers all cases when no improvement took place between some two iterations - t and $t + \Delta t_h$, in an amount of time - Δt_h. It was decided that the best method to detect stagnation is to measure the Euclidean distance between the current fitness of best particle at iterations $t+\Delta t_h$ and t and decide if the result is a relative improvement compared to a fraction (ε) of the best particle's fitness at iteration t :

$$\left\|\hat{y}_{t+\Delta t_h}-\hat{y}_t\right\|<\varepsilon\cdot\left\|\hat{y}_t\right\|. \tag{4.15}$$

4.3.2. RS-PSOD Operator

The **riot-when-stagnation** (RS-PSOD) operator was built to help PSO resolve stagnation states. It imitates social rioting: while the swarm is in stagnation a high number of particles from the current iteration do not follow the best particle in their group anymore and manifest extreme disagreements similar to D_e from eq. (4.9). However, the probability that such riots occur should be reduced if the execution of the current run is approaching to an end, otherwise too much randomness can be infused at a too later stage and the swarm may never converge or may impair a good solution. Therefore a linearly decreasing allocation scheme was considered for this type of disagreement operator.

The RS injector is defined as follows:

Definition 19. *Let P be a particle swarm optimization algorithm that has a social component. The function that injects in P a set of disagreements - Δ_{RS}, with an apply rule - ρ_{RS}, and transforms it into a particle swarm optimization with disagreements following the RS rule, namely a RS-PSOD algorithm - P_{RS}, is defined as:*

$$\Psi_{RS-PSOD}(P)=\Psi_{PSO}(P,\rho_{RS})=P_{DRS}. \tag{4.16}$$

The updating principle for RS-PSOD becomes (from eq. (4.4)):

$$\begin{aligned}X(t+1,x_i)&=\rho_{RS}(\Delta_{RS},\beta_i,t)\\&=\rho_{RS}(\Delta_{RS},X(t,x_i)+V(t,i)+C(t,x_i,y_i)+S(t,x_i,\hat{y}_i)+\zeta,t)\\&=X(t,x_i)+V(t,i)+C(t,x_i,y_i)+D_{RSi}(S(t,x_i,\hat{y}_i))+\zeta,\\C(t,x_i,y_i)&\to y_i,S(t,x_i,\hat{y}_i)\to D_{RSi}(\hat{y}_i),D_{RSi}\in\Delta_{RS},\forall i\in\overline{1,s}.\end{aligned} \tag{4.17}$$

The set of disagreements - D_{RSi}, consists of:

$$\Delta_{RS}=\{\varnothing_D,D_{RS}\}. \tag{4.18}$$

D_{RS} is an extreme disagreement, a generalization of D_e from eq. (4.9). It represents the individuals that are rioting against the status-quo in periods of crisis. Mathematically, it is a Hadamard product between the social update behavior (a subcomponent of β_i from eq. (4.2)) that is the social component S, and a vector r containing random uniformly distributed values in the intervals $[-\lambda_u,-\lambda_l]$ and $[+\lambda_l,+\lambda_u]$, with $\lambda_l,\lambda_u\in\mathrm{R}_+^*,\lambda_u>\lambda_l$ and $\lambda_l\geq 1$:

$$D_{RS}(S)=r_i\otimes S,r_i=r_{i_1}+sgn(r_{i_1})\cdot\lambda_l,r_{i_1}\sim\mathrm{U}(-(\lambda_u-\lambda_l),+(\lambda_u-\lambda_l)), \tag{4.19}$$

where r_i is the i-th component of r and r_{i_1} is a random number for each r_i.

Definition 20. *Let $\theta_{RS}(t,i)\sim U(0,1)$ be an uniformly distributed random variable that is generated at each iteration t for each particle i. Let*

$\delta = \dfrac{t}{t_{max}} \in [0,1]$ *be the current execution progress indicator, where* t_{max} *is the total number of iterations. The RS-PSOD operator is defined as follows:*

$$\rho_{RS}(\Delta_{RS},\beta_i,t) = \begin{cases} \varnothing_D(S), & \text{if } \theta_{RS}(t,i) < \delta \text{ or } (4.15) \text{ is false} \\ D_{RS}(S), & \text{if } \theta_{RS}(t,i) \geq \delta \text{ and } (4.15) \text{ is true} \end{cases} . \quad (4.20)$$

Compared to 6σ-PSOD operator, that injects disagreements based on a Gaussian distribution at all iterations, the RS-PSOD operator triggers riots only when stagnation occurs.

4.3.3. Experimental Setup

For testing the RS-PSOD algorithms against classical configurations a similar test environment to the one used for 6σ-PSOD was set up. After rigorous tuning, the best results for the extreme disagreement were obtained for the following configuration: $\lambda_l = 1, \lambda_u = 2, \Delta t_h = 5$ with a relative stagnation threshold of 0.005 in most cases (depending on the particularities of the given problem). It can be noticed that the extreme disagreement is the same as D_e from eq. 4.9. Truly that is the value which yielded the best experimental results.

The functions whose results are provided in the next subsection are the most representative examples from the whole set of 10 functions where stagnation can occur, grouped by categories: a plateau function (LF_1), a shifted highly multi-modal CEC 2005 function (LF_4) and a regular highly multi-modal function (LF_8).

4.3.4. Results

A Plateau Function

As it can be observed from Table 4.13, there is only one case (for PSO-VG with 50 particles in 30 dimensions) in which the RS approach failed to provide better performance.

The number of riots indicate how many stagnation situations were detected and handled on average in a run using the RS operator. In all tests there are higher values for 25 particles because it is harder to accomplish the proposed improvement amount with less particles. On LF_1 there is a high likelyhood of stagnation because of its large plateau.

Table 4.13: RS-PSOD results for LF_1.

algorithm	d	s	mean	std. dev.	median	riots
SPSO	30	25	7145.40	9083.21	3859.55	0.00
SPSOD–RS	30	25	3066.02	12126.44	289.80	102.35
PSO–VG	30	25	127751.53	227428.89	32393.18	0.00
PSO–VGD–RS	30	25	38663.99	106789.72	16691.89	111.00
SPSO	30	50	462.57	1677.74	81.32	0.00
SPSOD–RS	30	50	174.63	488.72	83.13	41.01
PSO–VG	30	50	4909.38	13650.26	896.37	0.00
PSO–VGD–RS	30	50	5837.03	9704.83	2003.35	43.23
SPSO	50	25	65604.93	110255.89	35342.06	0.00
SPSOD–RS	50	25	22222.69	44972.17	9641.15	110.12
PSO–VG	50	25	658684.63	466908.89	521877.29	0.00
PSO–VGD–RS	50	25	371364.13	396629.85	221172.45	125.79
SPSO	50	50	9708.33	41110.93	1788.16	0.00
SPSOD–RS	50	50	4091.20	8789.44	1048.60	31.08
PSO–VG	50	50	140868.25	222987.33	66829.56	0.00
PSO–VGD–RS	50	50	132610.32	193535.20	69786.47	38.91

In all sixteen situations (25 and 50 particles in 30 and 50 dimensions), SPSOD-RS is providing a consistent performance enhancement. Only with a single exception, 50 particles in 30 dimensions, the same is valid for PSO-VGD-RS.

As it can be seen in the graphs from Fig. 4.16 and Fig. 4.17 for the case of SPSO-RS vs. SPSO with 25 particles in 50 dimensions, both the best and the worst individual have a better evolution when the RS variants are used. As for the population average distance for SPSOD-RS, that is slightly increased by the injection of extreme disagreements only when needed, as shown in Fig. 4.18. There is not a constant process that injects disagreements like in the case of the 6σ-PSOD operator and the local exploitation is not enhanced at all using this technique. Yet, for plateaus, that do not need a fierce local exploitation, the results are far better for RS than in the case of the classical PSO configurations in test.

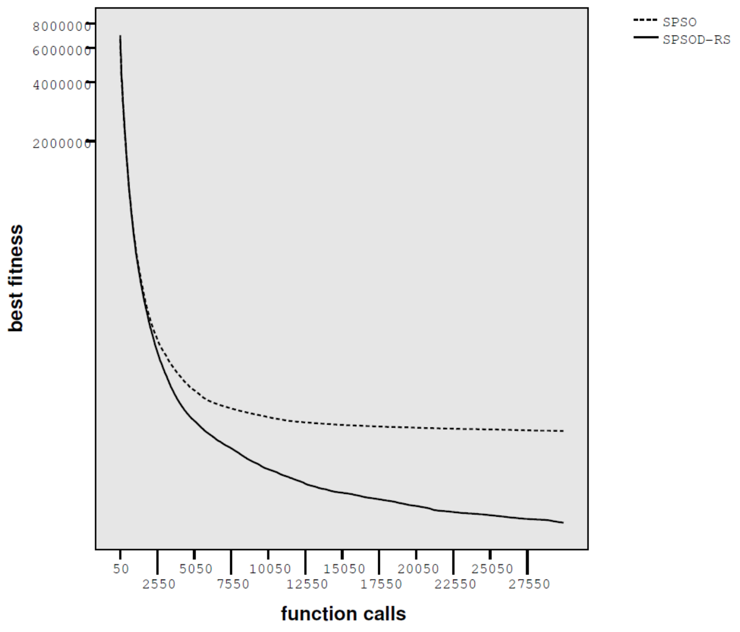


Figure 4.16: SPSOD-RS vs. SPSO best fitness graph with 25 particles for LF_1 in 50 dimensions.

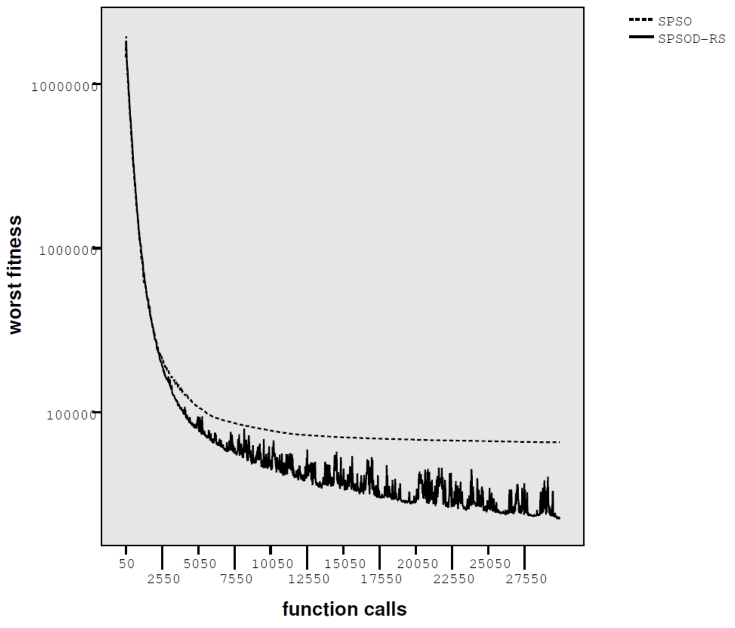


Figure 4.17: SPSOD-RS vs. SPSO worst fitness graph with 25 particles for LF_1 in 50 dimensions.

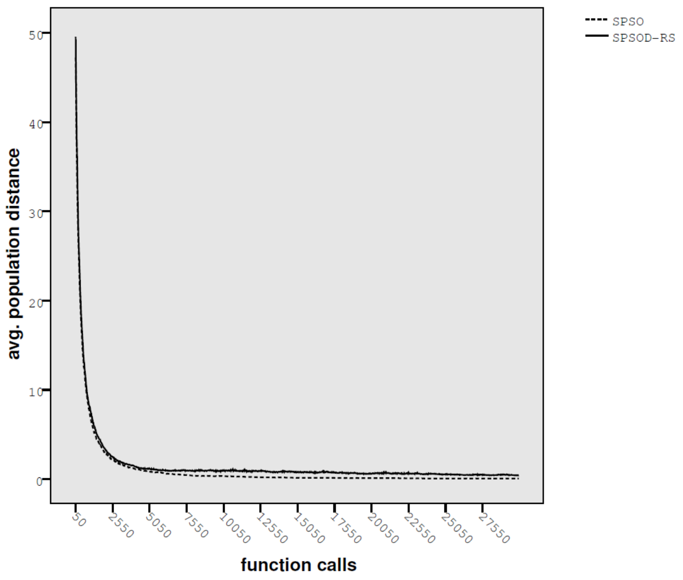


Figure 4.18: SPSOD-RS vs. SPSO avg. pop. dist. graph with 25 particles for LF_1 in 50 dimensions.

A Shifted Multi-Modal Function

For testing the RS approach on a shifted multi-modal function, LF_4 was picked from the set of CEC 2005 test functions.

Table 4.14 shows the results when running PSO algorithms with the riot-when-stagnation feature versus their original counterparts. The column showing the mean fitness contains lower values for RS-PSOD algorithms, which means that they provide a better performance and it is better to use them. For the cases with 25 particles there were approximately 170 riots needed per run and for the cases with 50 particles, when it is easier to grow gradually in performance across iterations, around 60 riots.

For shifted multi-modal test functions the gain is not so spectacular as in the case of the plateau functions, but it demonstrates empirically that even on most disturbed and multi-modal environments the disagreements metaphor deals well with the considered problems. The robustness of the original algorithms is preserved and the convergence speed, the population variety and the overall performance are increased.

A Regular Multi-Modal Function

When the RS approach was tested on a regular multi-modal function, on Ackley's function (F_8), the improvement ratio seen in the results is approximately the same as in the above described shifted case.

The results can be analyzed in Table 4.15 and they show that there is a slight but genuine improvement in performance for the disagreements enabled variants of PSO. Disagreements are acting to repair stagnations in swarm.

Table 4.14: RS-PSOD results for LF_4.

algorithm	d	s	mean	std. dev.	median	riots
SPSO	30	25	141.70	37.73	139.14	0.00
SPSOD–RS	30	25	126.73	33.71	123.69	174.49
PSO–VG	30	25	223.58	47.31	218.86	0.00
PSO–VGD–RS	30	25	199.43	52.68	196.03	170.48
SPSO	30	50	77.05	21.75	72.54	0.00
SPSOD–RS	30	50	76.64	21.21	72.70	64.43
PSO–VG	30	50	132.98	32.78	127.52	0.00
PSO–VGD–RS	30	50	128.14	35.03	125.60	64.62
SPSO	50	25	309.46	60.31	313.20	0.00
SPSOD–RS	50	25	285.64	61.28	284.00	167.71
PSO–VG	50	25	493.50	86.74	487.96	0.00
PSO–VGD–RS	50	25	459.00	71.82	450.00	171.76
SPSO	50	50	175.38	41.06	171.86	0.00
SPSOD–RS	50	50	170.10	36.76	168.15	48.29
PSO–VG	50	50	313.71	62.58	312.94	0.00
PSO–VGD–RS	50	50	306.12	69.72	298.52	58.66

Table 4.15: RS-PSOD results for LF_8.

algorithm	d	s	mean	std. dev.	median	riots
SPSO	30	25	11.54	2.86	11.75	0.00
SPSOD–RS	30	25	10.27	2.71	9.98	177.63
PSO–VG	30	25	17.68	2.13	18.55	0.00
PSO–VGD–RS	30	25	17.66	1.78	18.28	184.07
SPSO	30	50	4.12	2.49	3.98	0.00
SPSOD–RS	30	50	3.10	1.77	2.51	56.64
PSO–VG	30	50	11.01	3.77	10.47	0.00
PSO–VGD–RS	30	50	10.76	3.66	10.34	65.55
SPSO	50	25	15.10	2.02	15.36	0.00
SPSOD–RS	50	25	13.67	2.12	13.90	169.89
PSO–VG	50	25	19.31	0.47	19.42	0.00
PSO–VGD–RS	50	25	19.17	0.58	19.30	188.89
SPSO	50	50	8.22	2.77	7.74	0.00
SPSOD–RS	50	50	7.61	2.53	7.23	43.56
PSO–VG	50	50	16.60	2.12	17.13	0.00
PSO–VGD–RS	50	50	16.23	2.54	16.77	72.28

Comparison with 6σ-PSOD

RS-PSOD provides better performance compared to PSOs that have no disagreements, but the question is how it stands in terms of performance against

6σ -PSOD. The short answer is that it was experimentally determined that *6σ* -PSOD performs better.

In this section LF_8 in 30 dimensions is taken into discussion. Table 4.16 provides the race results between *6σ* -PSOD vs. RS-PSOD.

For 25 particles the results are slightly better for *6σ* . Even with around 180 riots and around 3.75 extreme individuals per run, they cannot beat the *6σ* approach which although it acts blindly, it simulates what happens in nature and with around 3.4 partial disagreements and 0.085 extreme disagreements per run it outperforms RS-PSOD.

For 50 particles in swarm, the results are in *6σ* -PSOD's favour again. For SPSOs' case the situation is clearer, the difference for the mean best fitness, its standard deviation and the median is bigger than in the other cases.

For Standard PSO some graphs are provided: Fig. 4.19 is the best fitness graph, Fig. 4.20 is the graph for the evolution of the worst individual in the swarm and Fig. 4.21 shows the population average distance between individuals. The graph with the best fitness evolution shows how *6σ* is heading better to convergence than the RS approach and the graph with the worst individuals is showing that the least performant in *6σ* -PSOD is better than the one in RS-PSOD. Y-axis values for the graph of population diversity are plotted this time on a logarithmic base 10 scale in order to be able to emphasize the details. In an initial phase, from 0 to 6000 fitness function calls, the *6σ* assures a more diversified population, then in a later phase, after 6000, the RS method takes the lead towards 20000 function evaluations. If the surface inbetween the 2 plots is measured for these two phases the area for the first phase is greater than the area for the second phase. Meanwhile, the phase 2 is longer and happens later. From here it can be concluded that *6σ* is providing a higher amount of diversity in the beginning phase and the RS provides a lower diversity at a later stage because then it comes the time when stagnation occurs. From experimental experience and the design of many evolutionary algorithms like simulated annealing it is generally better to have a high diversity at the beginning of the optimization and a lower one at the end. That is why *6σ* does better, but RS does better than the original algorithms too.

Table 4.16: *6σ* -PSOD vs. RS-PSOD results for LF_8 in 30 dimensions.

algorithm	**s**	**mean**	**std. dev.**	**median**	**p.d.i.**	**e.d.i.**	**riots**
SPSOD$_{6\sigma}$	25	9.88	2.97	9.38	3.32	0.08	0.00
SPSOD–RS	25	10.27	2.71	9.98	0.00	3.70	177.63
PSO–VGD–RS	25	17.66	1.78	18.28	0.00	3.83	184.07
PSO–VGD$_{6\sigma}$	25	16.84	2.52	17.52	3.50	0.09	0.00
SPSOD$_{6\sigma}$	50	1.57	1.30	1.34	7.17	0.20	0.00
SPSOD–RS	50	3.10	1.77	2.51	0.00	4.72	56.64
PSO–VGD–RS	50	10.76	3.66	10.34	0.00	5.46	65.55
PSO–VGD$_{6\sigma}$	50	9.41	4.33	8.49	6.82	0.18	0.00

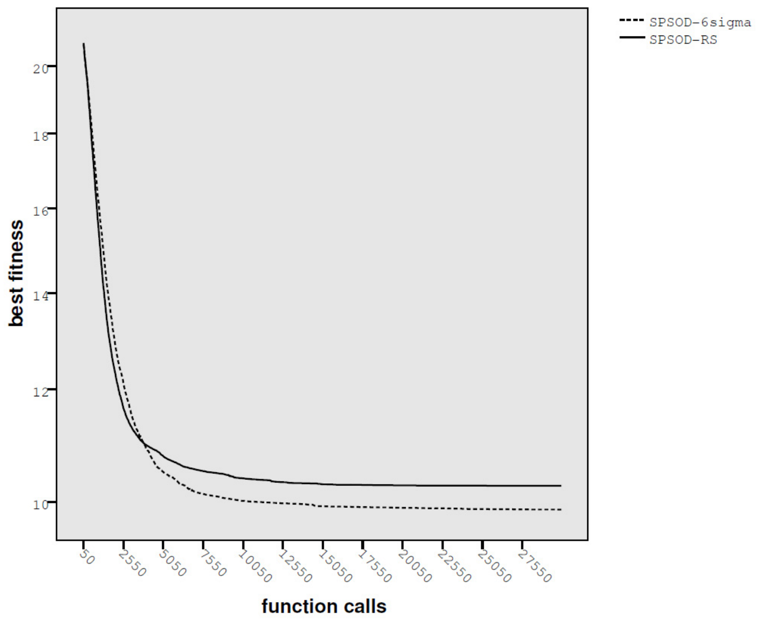


Figure 4.19: SPSOD-RS vs. $SPSOD_{6\sigma}$ best fitness graph with 25 particles for LF_8 in 30 dimensions.

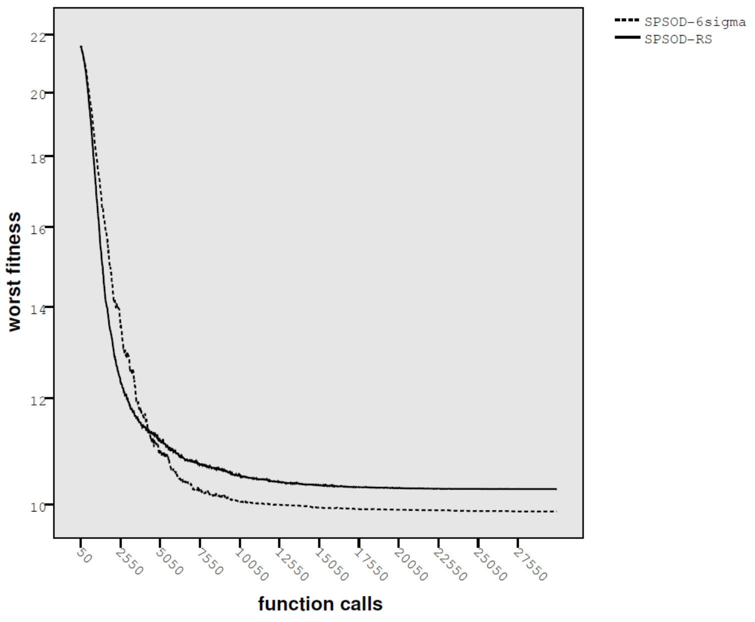


Figure 4.20: SPSOD-RS vs. $SPSOD_{6\sigma}$ worst fitness graph with 25 particles for LF_8 in 30 dimensions.

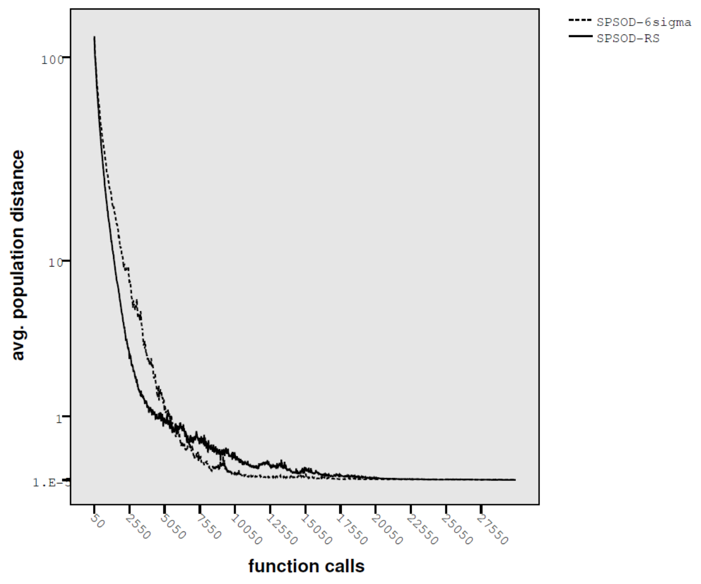


Figure 4.21: SPSOD-RS vs. $SPSOD_{6\sigma}$ avg. pop. dist. graph with 25 particles for LF_8 in 30 dimensions.

4.3.5. Concluding Remarks

The conclusion can only be very short: **RS-PSOD was conceived as a repair method. On the other hand, *6σ* -PSOD was built as a holistic approach to PSO. Both outperform classical PSOs in terms of performance without significant computational overhead.**

RS-PSOD is a very handy and easy method to regain the swarm's ability to discover better positions in the search space, proving that disagreements are good as a repair method also. Using only extreme disagreements when needed (when stagnation occurs) the swarm's evolution is revived. The same thing happens in society during periods of crisis - the before behaviors are challenged and completely new ideas come into play.

The 6σ approach is directing the swarm towards better solutions as a holistic approach. Its drive comes into effect at every iteration. The search process is shaped like a society with disagreeing individuals here and there. Using relevant benchmarks it was proved that the *6σ* approach provides better results than RS-PSOD in tests and it has less parameters than RS. However, RS-PSOD is intervening in a middle or later phase when no significant improvements take place, it is not a shaping but rather a repair method that does what it is supposed to do: artificially bails out the swarm from its performance crisis.

The experimental results provided for both *6σ* -PSOD and RS-PSOD are a successful proof of concept for disagreements in PSO. It can be easily applied to other swarm intelligence algorithms also. What is more important is that now the door is open for applying disagreements in other evolutionary algorithms.

5. GENETIC ALGORITHMS WITH DISAGREEMENTS

5.1. Disagreements - An Attitude for GAs

5.1.1. Concepts

Disagreements are a phenomenon that can be well understood and studied in the context of social interactions. PSO and other swarm intelligence that have a social component can use them in their advantage. Nevertheless, when it comes to other evolutionary algorithms, like genetic algorithms, that do not posses social mechanisms in their fabric, it is hard at first to think of a way disagreements can be acclimatized into these algorithms. With the advent of *memetic algorithms (MAs)* in which individuals from a classical GA could develop a personality, the idea of disagreements inside a GA is plausible. While a memetic algorithm is adding a local search to a GA, a genetic algorithm with disagreements (GAD) is adding disagreements to a GA. In both cases, individuals get an attitude.

The definition that provides the way disagreements are injected into a real-valued GA and mirrors Definition 15 is provided below:

Definition 21. *Let G be a real-valued GA and ρ a disagreements apply rule. A* ***real-valued genetic algorithm with disagreements (GAD)****, G_D, is obtained by modifying G's updating principle with the rule ρ, as described by the following disagreement injector function:*

$$\Psi_{GA}(G,\rho) = G_D. \qquad (5.1)$$

5.1.2. Implementations

In the experiments related to GAD provided in this thesis disagreements were designed as a better mutation operator for real-valued GAs. As it will be shown, replacing the mutation operator with disagreements improves the overall performance of a GA and wins the advantages of a memetic algorithm. Therefore, GADs can be seen as:

- as a new class of memetic algorithms without a local search method or
- as genetic algorithms with a better mutation or
- as a class of their own kind.

Various implementations can drive GADs into one of the above three categories (or maybe into other categories also), but as a concept disagreements in genetic algorithms represent a different class of *GAs with attitude*.

The number of possible implementations of disagreements in GAs can be very high. In the next part of this thesis only a particular implementation, the *6σ*-GAD, that resembles *6σ*-PSOD is presented as *a proof of concept of disagreements in real-valued GAs*. Of course, there is also possible to build an RS-GAD approach similar to RS-PSOD, but because that would imply a repetition and would bloat the content of the thesis, it is not included.

5.2. 6σ-GAD Operator

5.2.1. Definition

In [61] the author introduces for the first time the notion of genetic algorithms with disagreements as a new mutation operator, modeled around the assumption that in an iteration there are some individuals that partially disagree and a very small minority that extremely disagree. The new mutation type acts in the same manner as the 6σ-PSOD operator. It is taken into consideration in order to prove that the idea of disagreements work also for other evolutionary algorithms outside the swarm intelligence category.

Since good results are obtained using PSO-VG, an algorithm where disagreements impact a larger part of the updating principle β_i, the 6σ-GAD operator is developed to mirror the 6σ-PSOD operator. In terms of (5.1) its definition is the following:

Definition 22. *Let G be a real-valued genetic algorithm. The function that injects in G the set of disagreements - $\Delta'_{6\sigma}$, with an apply rule - $\rho'_{6\sigma}$, and transforms it into a real-valued genetic with disagreements (GAD) following the $\rho'_{6\sigma}$ rule, namely a 6σ-GAD algorithm - $G_{D6\sigma}$, is defined as follows:*

$$\Psi_{6\sigma-GAD}(G) = \Psi_{GA}(G, \rho'_{6\sigma}) = G_{D6\sigma}, \tag{5.2}$$

For an individual from the population the update principle is represented by the iterational loop from the Bäck's evolutionary framework presented in Pseudocode 1. At an iteration t, the update principle yields for an individual i a behavior β_i, composed of the changes required by recombination, mutation and selection. In 6σ-GAD's update principle the standard mutation is replaced by the mutation that is represented by the disagreements with their rules. For 6σ-GADs the apply rule $\rho'_{6\sigma}$ acts upon a part of β_i which is the individual z after the step of recombination.

The employed subset of disagreements resemble those from [58]:

$$\Delta'_{6\sigma} = \left\{\varnothing_D, D'_p, D'_e\right\} \tag{5.3}$$

They are defined as follows:

Definition 23. *Let $\lambda_l, \lambda_u \in \mathrm{R}^*_+, \lambda_l < \lambda_u$. Let a and b be the vectors containing the lower and respectively, the upper bounds of the search space H^n. The partial disagreement - D'_p is defined as:*

$$D'_p(z) = z + q_i \otimes \frac{b-a}{2}, q_i \sim \mathrm{U}(-\lambda_l, +\lambda_l), z \in H^n, \tag{5.4}$$

where q_i is the i-th component of q and z is the changed individual after recombination from Pseudocode 1.

Definition 24. *Let $\lambda_l, \lambda_u \in \mathbb{R}_+^*, \lambda_l < \lambda_u$. Let a and b be the vectors containing the lower and respectively, the upper bounds of the search space H^n. The extreme disagreement D_e' is defined as:*

$$D_e'(z) = z + w_i \otimes \frac{b-a}{2}, w_i = w_{i_1} + sgn(w_{i_1}) \cdot \lambda_l, \tag{5.5}$$
$$w_{i_1} \sim \mathrm{U}(-(\lambda_u - \lambda_l), +(\lambda_u - \lambda_l)),$$

where w_i is the i-th component of w, w_{i_1} is an uniformly distributed random number generated each time for w_i and z is the changed individual after recombination from Pseudocode 1.

It can be easily noticed that both disagreements are the GA generalized counterparts of the ones defined for 6σ-PSOD.

5.2.2. Design Philosophy

For GAs, like for PSO, disagreements were designed to provide better neighborhood exploitation through D_p' and enhanced exploration using amplified values generated by D_e'. The concept is depicted in Fig. 5.1.

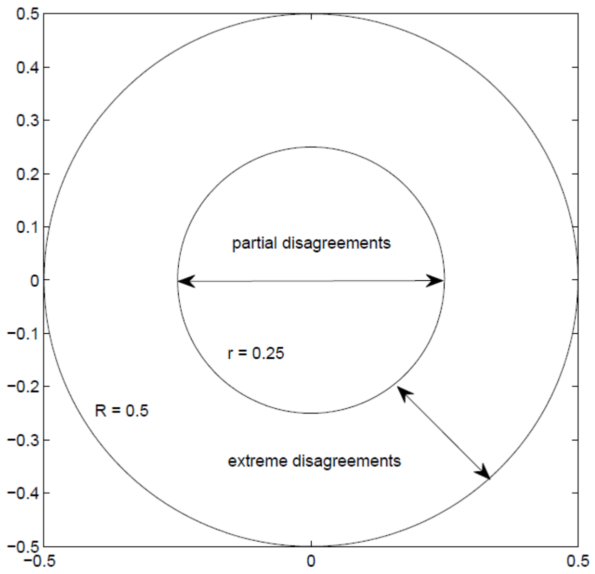


Figure 5.1: Disagreements for real-valued genetic algorithms.

"The concept of partial and extreme disagreements is related to neighborhoods. Partial disagreements are values in the vicinity of the original value, while extreme disagreements are either more distant values or opposite ones.

As a further explanation, if one may think of the alphabet, some partial disagreements for letter M can be the letters N, O, P or L (a radius of partial disagreement should be defined), while some extreme disagreements for letter A can be Z (the dichotomy ''first-last letter'', although it can also be partial disagreement if we consider a circular alphabet) or even a non-letter, like ''!''; also, for letter A, an extreme disagreement can be defined as a more distant character, like H. This example is given to show that genetic algorithms with disagreements can be used also on discrete domains.'' ([61])

5.2.3. Selector Function

The selector function resembles the one for 6σ-PSOD, but here it is defined without filtering. Therefore, at each iteration t, for each individual i, $\theta(t,i) \sim \mathrm{N}(\mu_{6\sigma}, \sigma_{6\sigma}^2)$ is generated - a Gaussian distributed random variable with a chosen mean $\mu_{6\sigma}$ and a standard deviation $\sigma_{6\sigma}$. Based on (63), (64), (65) and the Gaussian regions defined in formulae (50), (51) and (52), the selector function for GADs is defined as:

$$\rho'_{6\sigma}(\Delta'_{6\sigma}, \beta_i, t) = \begin{cases} \varnothing_D(z), & \text{if } \theta(t,i) \in R_{1,2\sigma} \\ D'_p(z), & \text{if } \theta(t,i) \in R_{3,4\sigma} \\ D'_e(z), & \text{if } \theta(t,i) \in R_{5,6\sigma} \end{cases}, \tag{5.6}$$

where z is the individual at iteration t after recombination. The rule is supposed to replace the mutation and it is noteworthy that in the case of *"no-op"* no mutation takes place. Compared to 6σ-PSOD, this is a more invasive method. Its design was chosen not to mirror 100% the one provided for PSO in order to explore another facet of how disagreements can be used in optimization.

5.2.4. Experimental Setup

The experimental part for proving the effectiveness of injecting disagreements in real-valued genetic algorithms was conducted using the established rules in 3.3.3 and in similar conditions with experiments related to 6σ-PSOD. In all cases presented here, a small population of 50 individuals was used. The interest goes in comparing real-valued genetic without elitism (GA) and with elitism (eGA) versus their 6σ-GAD enhanced counterparts. All algorithms used BLX-α crossover ($\alpha = 0.5$) with probability 0.7. Original GAs used a Mühlenbein mutation with a probability of 0.1.

After calibration, the following values for the parameters of 6σ-GAD were found to work in most cases: $\lambda_l = 0.25, \lambda_u = 0.5$, therefore GADs shown in tables have that settings.

The output variables are the same as in the case of 6σ-PSOD experiments: mean of best fitness and its standard deviation, the median of best fitness and the average number of disagreements in an iteration (partial and extreme). Other interesting outputs such as convergence, worst fitness and population distance graphs and for brevity are sparingly presented.

Results on some relevant test functions are provided.

5.2.5. Experimental Results

A Plateau Function

For LF_1 in 30 dimensions, using a GAD without elitism against a similar GA, the mean fitness is 16.4 times closer to the global optimum. When using elitism the advantage is somewhat similar, around 14.40 times better when using eGAD.

In 50 dimensions it is even better: the mean fitness is 1814% and 1372% closer to the solution for GAD vs. GA, and eGAD vs. eGA, respectively. Table 5.1 shows a detailed performance overview.

Graphs are provided for the case with 50 dimensions. Similar to 6σ -PSODs, GADs exhibit a greater population diversity and a better coverage of the search space (see Fig. 5.4). It seems that the worst individual from the GAs (GA and eGA) is better than the worst individual from the GADs (see Fig. 5.3). However this doesn’t impede the GADs' higher capability of convergence, as depicted in Fig. 5.2.

When seeking for improved performance on plateau functions like LF_1 it is a good idea to inject some amount of disagreements, both partial and extreme.

Table 5.1: 6σ -GAD results for LF_1.

algorithm	d	mean	std. dev.	median	p.d.i.	e.d.i.
GA	30	5804.16	6329.69	3639.15	0.00	0.00
$GA_{6\sigma}$	30	353.72	175.07	306.59	13.52	2.26
eGA	30	7433.49	7174.22	4899.34	0.00	0.00
$eGA_{6\sigma}$	30	515.87	220.13	482.68	13.51	2.26
GA	50	35832.31	29455.30	27212.73	0.00	0.00
$GA_{6\sigma}$	50	1974.45	769.92	1881.21	13.56	2.27
eGA	50	42084.21	37268.62	33932.76	0.00	0.00
$eGA_{6\sigma}$	50	3065.40	1243.48	2899.27	13.54	2.26

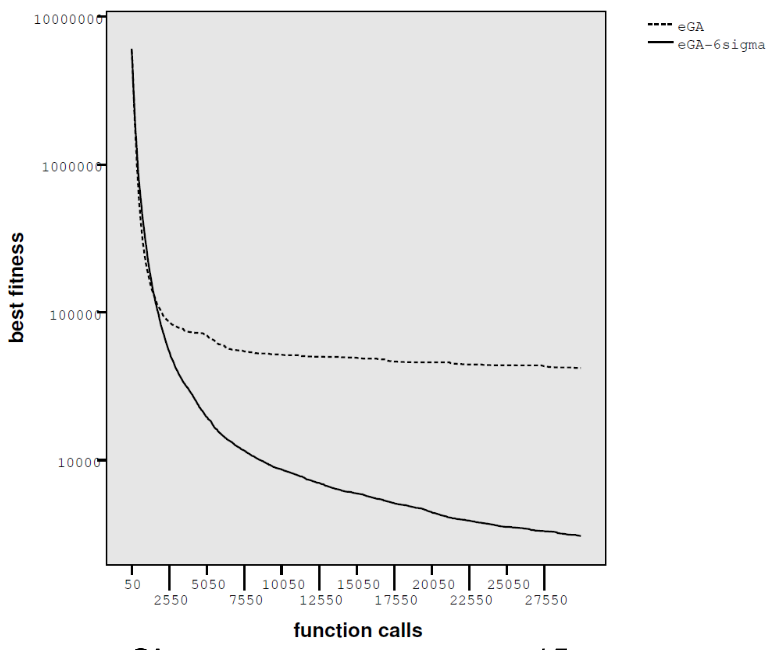


Figure 5.2: $eGA_{6\sigma}$ vs. eGA best fitness graph for LF_1 in 50 dimensions.

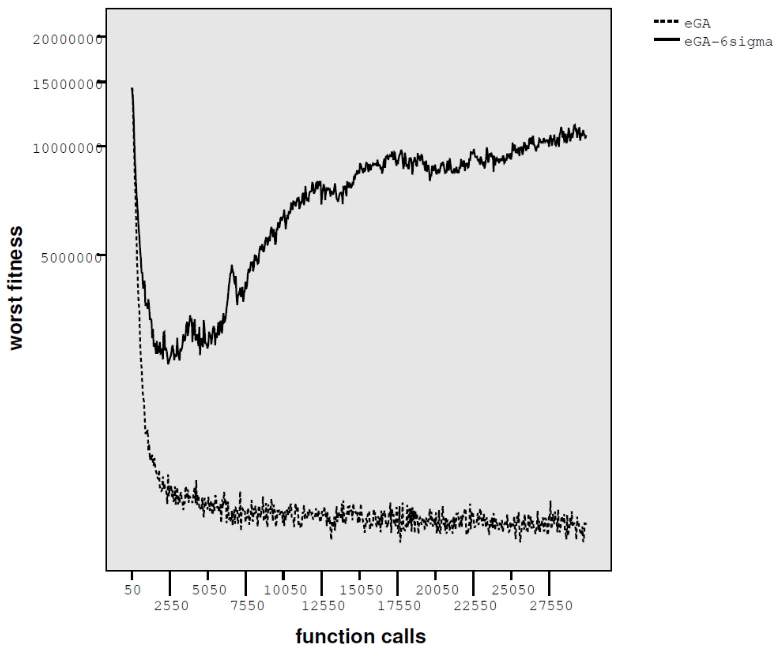


Figure 5.3: $eGA_{6\sigma}$ vs. eGA worst fitness graph for LF_1 in 50 dimensions.

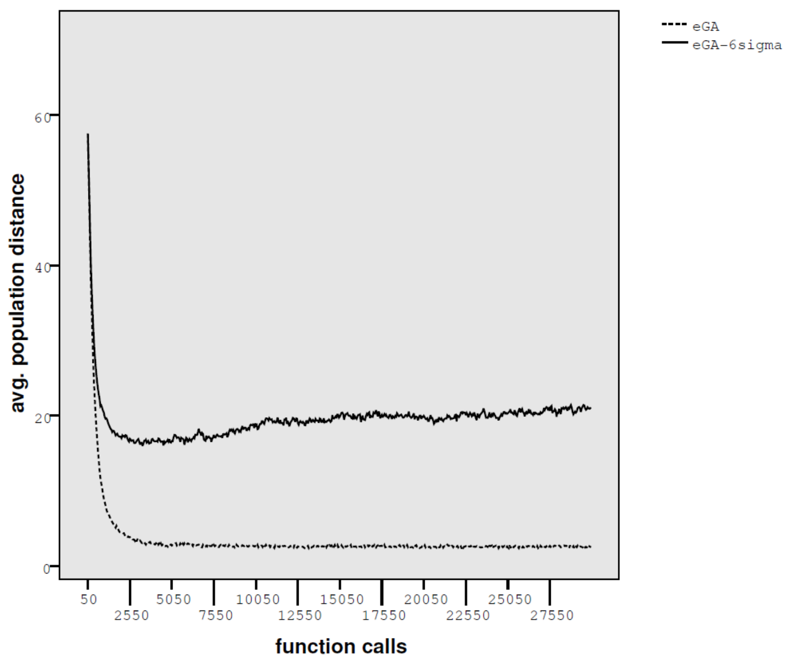


Figure 5.4: $eGA_{6\sigma}$ vs. eGA avg. population distance graph for LF_1 in 50 dimensions.

A Shifted Multi-Modal Function

Table 5.2 shows the results for one of the CEC 2005 test functions, Shifted Rastrigin (LF_4). The results prove that in most cases in which disagreements are used there is a slight better outcome. However, the improvement is marginal in this case, so it might not be worthy to use disagreements in this case.

Table 5.2: 6σ -GAD results for LF_4 .

algorithm	d	mean	std. dev.	median	p.d.i.	e.d.i.
GA	30	243.08	34.68	236.79	0.00	0.00
$GA_{6\sigma}$	30	241.24	17.48	241.23	13.55	2.25
eGA	30	272.10	34.47	271.63	0.00	0.00
$eGA_{6\sigma}$	30	270.74	18.50	272.42	13.54	2.26
GA	50	570.72	61.29	567.05	0.00	0.00
$GA_{6\sigma}$	50	556.99	32.93	557.60	13.54	2.27
eGA	50	595.05	46.63	593.79	0.00	0.00
$eGA_{6\sigma}$	50	598.71	25.59	601.51	13.53	2.26

A Low Dimensional Problem

Himmelblau is a perfect test case for low dimensional search spaces. From Table 5.3 only one output variable is very important: the convergence rate, which is higher for disagreements enabled algorithms.

Table 5.3: 6σ -GAD results for LF_6 .

algorithm	conv. rate	mean	std. dev.	median	p.d.i.	e.d.i.
GA	0.90	0.0005	0.0026	0.00	0.00	0.00
$GA_{6\sigma}$	1.00	0.0000	0.0000	0.00	13.52	2.26
eGA	0.99	0.0004	0.0017	0.00	0.00	0.00
$eGA_{6\sigma}$	1.00	0.0000	0.0000	0.00	13.57	2.26

A Regular Multi-Modal Function

In order to illustrate the convergence behavior of the 6σ -GADs, Griewank (LF_7) is used here as a test case. From Table 5.4 it can be concluded that GADs bring significant improvement in highly multi-modal environments, one of the very important criterion for modern optimization techniques.

Table 5.4: 6σ -GAD results for LF_7 .

algorithm	d	mean	std. dev.	median	p.d.i.	e.d.i.
GA	30	7.95	2.13	7.61	0.00	0.00
$GA_{6\sigma}$	30	3.93	0.79	3.98	13.54	2.26
eGA	30	9.27	2.07	9.38	0.00	0.00
$eGA_{6\sigma}$	30	4.55	0.63	4.53	13.55	2.26
GA	50	10.84	1.97	10.95	0.00	0.00
$GA_{6\sigma}$	50	6.33	0.97	6.41	13.56	2.26
eGA	50	12.09	2.12	12.13	0.00	0.00
$eGA_{6\sigma}$	50	7.30	0.80	7.42	13.54	2.27

The best fitness graph for eGA vs. eGAD in 30 dimensions is shown in Fig. 5.5. One can see that eGAD is taking early the lead and clearly finishes with a very good fitness value.

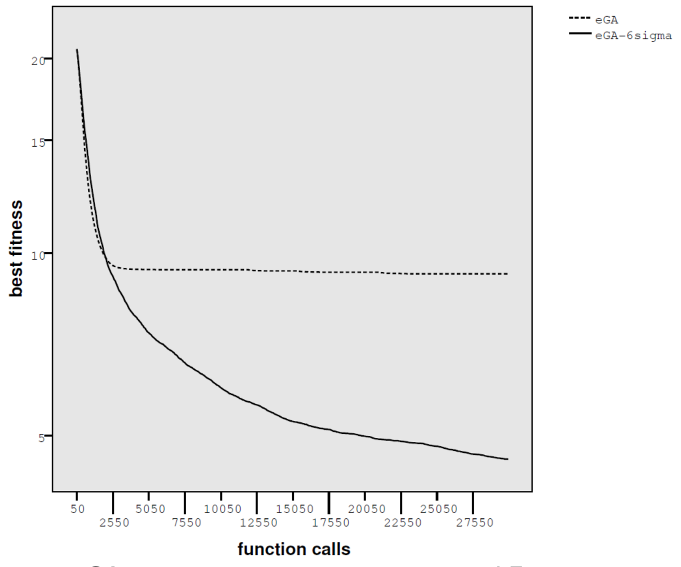


Figure 5.5: $eGA_{6\sigma}$ vs. eGA best fitness graph for LF_8 in 30 dimensions.

The graph of the worst graph individual is proving once more time that in GADs the worst fitness is declining with the run's length. However, this does not affect the evolution of the best individual. The average distance between members of the swarm is higher for GADs than in the case of their analogous GAs, as depicted in Fig. 5.7.

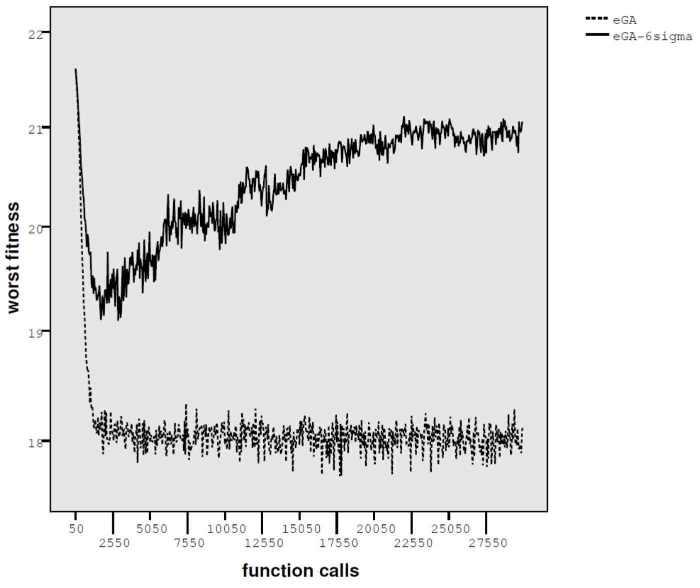


Figure 5.6: $eGA_{6\sigma}$ vs. eGA worst fitness graph for LF_8 in 30 dimensions.

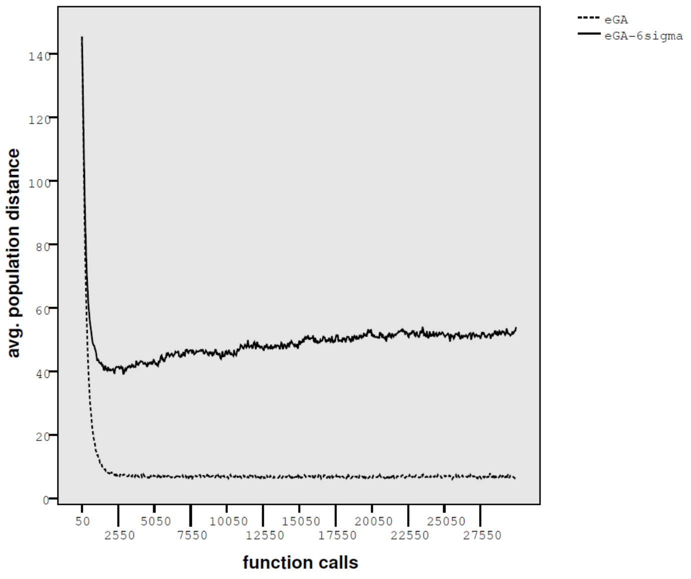


Figure 5.7: $eGA_{6\sigma}$ vs. eGA avg. population distance graph for LF_8 in 30 dimensions.

5.2.6. Disagreements vs. Mutation

To assess the difference between the disagreements mutation operator and a regular Gaussian real-valued mutation, two test cases were considered: one for *LF_1* and another one for *LF_7*, both cases in 50 dimensions and with GAs with elitism (eGAs). The rate of mutation in GADs is around 35 %, therefore a high mutation eGA, meGA, with the same percentage of mutation as in GADs was tested against an eGAD. The results are provided in 5.5. They show without any doubt that using disagreements yields a better performance than ordinary Gaussian mutation.

Table 5.5: *6σ*-GAD vs. regular mutation results for *LF_7*.

algorithm	f	mean	std. dev.	median	p.d.i.	e.d.i.
meGA	LF_1	11980.41	15997.13	7456.13	0.00	0.00
eGA$_{6\sigma}$	LF_1	3065.40	1243.48	2899.27	13.54	2.26
meGA	LF_7	24.03	13.88	19.89	0.00	0.00
eGA$_{6\sigma}$	LF_7	9.96	3.55	9.67	13.54	2.26

5.3. Concluding Remarks

The results from testing *6σ*-GAD show that this new approach yields in better performance on most test cases by maintaining a diverse population. As in the case of *6σ*-PSOD, this is happening because the local exploitation is promoted by partial disagreements, while the exploration is increased using extreme disagreements.

Injecting disagreements in real-value GAs is beneficial when dealing with plateau and/or highly multi-modal functions. Excepting few cases the improvement is significant and the performance is enhanced as in the case of PSODs. This observation leads to the conclusion that disagreements may be a feature which any evolutionary algorithm can benefit from.

6. CONCLUSIONS

6.1. Concluding Remarks

In 2011 Time Magazine designated "The Protester" as The Person of the Year ([5]). It is not strange that the world is changing through disagreements and this thesis provided empirical evidence to support the introduction of this idea in designing evolutionary algorithms.

At first, the concept of disagreements applied in PSO yielded very good experimental results with the advent of 6σ-PSOD operator. The normally distributed 6σ scheme works wonderfully. The injection of partial disagreements increases the local exploitation and the injection of extreme disagreements strengthens the exploration. The results for the derived algorithms are without any doubt better. The extra added computational cost is minimal.

After the successful approach with 6σ-PSOD, it was shown that extreme disagreements can be used to resolve the stagnation in the swarms, by using the RS-PSOD operator. Tests have shown that the method is a reliable repair technique, yielding better outcomes. Still, from experimental data, it is recommended that the 6σ-PSOD should be used to derive superior algorithms, as it is a holistic approach, not a repair one, and has a better drive. As the old saying says: "An ounce of prevention is worth a pound of cure."

Good results obtained with the normally distributed disagreements on a social-only PSO, PSO-VG, on which the impact upon the updating principle is higher, opened the door for experimenting with algorithms outside the swarm intelligence area. Very good results in tests with real-valued genetic algorithms applying a 6σ rule, the 6σ-GAD mutation instead of the regular mutation, revealed that the new metaphor of disagreements definitively is worth of consideration for the larger category of evolutionary computation.

Overall, this thesis introduced the concept of disagreements in swarm intelligence through PSO and to a larger extent, in evolutionary computation, through real-valued genetic algorithms. It provided empirical data to prove that the new approach is yielding better performance.

6.2. Contributions

Based on the observations from nature and the promising provided experimental data, this work may open a new research area in the field of evolutionary computation: the use of disagreements to derive superior population-based algorithms with enhanced exploration and exploitation capabilities. This thesis is a proof of concept and it uses a specially crafted randomness injection into the updating principle of the evolutionary algorithms with a simple 2-layer neighborhood scheme in order to simulate the disagreements that naturally occur in-between members of a society or of a group.

In detail, the following contributions should be attributed to this thesis and the associated published work during the doctoral programme (all citations are referring to author's publications):

The disagreements metaphor. The new disagreements metaphor is based on real-world observations and applied to swarm intelligence and evolutionary computation algorithms. It is explained in detail and further developed in Section 1. Its introduction in PSO is done for the first time in [58]. The first generalization to EAs appears in [61].

A theoretical foundation for disagreements. In Section 2, a general theoretical foundation for implementing disagreements in EAs is provided.

A testing methodology for disagreements. A test methodology, named *race testing*, is provided in Section 3 to correctly asses the new concept of disagreements. Based on DACE, it can correctly compare the performance of the disagreements-derived algorithms with their original versions.

PSO with disagreements. Disagreements are introduced as a new social behavior in PSO, thus affecting only the social component of the updating principle (Section 1). This was also previously discussed in [58] and in [60].

6σ-PSOD. A promising implementation of disagreements in PSO, the *6σ*-PSOD operator, is provided in Section 2. It is a Gaussian scheme of injecting partial (to increase local exploitation) and extreme disagreements (to increase exploration) on a 2-layer neighborhood structure around the social component of the updating principle, that simulates real-life group opinions, first introduced in [58]. Empirical experimental data shows a great improvement in performance when used in derived new algorithms.

RS-PSOD. A repair method that proves the usefulness of applying extreme disagreements to mitigate stagnation, that is described in detail in Section 3. The concept was first published in [60].

GA with disagreements. Disagreements are introduced in real-valued GAs as an alternative to memetic algorithms, a new class of GAs: *GAs with attitude* in Section 1.

6σ-GAD. A *6σ* approach to real-valued GAs, that is shaped like a new mutation operator. It was first introduced in [61]. It is described in Section 2. The experimental data proves its superiority over regular GAs.

A proof of concept for disagreements. Except Chapter 2, which provides a succinct overview of the state of the art, all the other chapters in this thesis explain the concept of disagreements and prove that it is useful in optimization.

Needless to say, because the above contributions are at the core of the algorithms, their potential impact can affect all areas where evolutionary algorithms are used.

6.3. Future Work

There are several future research directions that can be pursued in the near future:

• the improvement of the current disagreements operators

• the development of new implementations of disagreements for PSO and real-valued GAs and other evolutionary computation algorithms

- checking the concept for other algorithms outside evolutionary computation
- studies regarding disagreements in social networks

An important idea that arises from this work is that exceptions from the rule, pictured here as disagreements, should be encouraged and accommodated when designing systems because they are a true source of natural change.

References

[1] D. Acemoglu, V. Chernozhukov, and M. Yildiz. Learning and disagreement in an uncertain world. NBER Working Papers 12648, National Bureau of Economic Research Inc., October 2006.

[2] D. Acemoglu, G. Como, F. Fagnani, and A. Ozdaglar. Opinion fluctuations and disagreement in social networks. CoRR, abs/1009.2653, 2010.

[3] D. Acemoglu, M.A. Dahleh, I. Lobel, and A. Ozdaglar. Bayesian learning in social networks. NBER Working Papers 14040, National Bureau of Economic Research Inc., May 2008.

[4] D. Acemoglu, A.E. Ozdaglar, and A. P. Gheibi. Spread of misinformation in social networks. CoRR, abs/0906.5007, 2009.

[5] K. Andersen. The person of the year. Time Magazine, December 2011. http://www.time.com/time/person-of-the-year/2011/. [Online; accessed 05-January-2012].

[6] R. Axelrod. The Dissemination of Culture: AModel with Local Convergence and Global Polarization. J. Conflict Resolut., 41(2):203–226, 1997.

[7] T. Bäck, D.B. Fogel, and Z. Michalewicz, editors. Evolutionary Computation 1: Basic Algorithms and Operators. Institute of Physics Publishing, Bristol,2000.

[8] V. Bala and S. Goyal. Learning from neighbours. Review of Economic Studies, 65:595–621, 1998.

[9] A. Banerjee and D. Fudenberg. Word-of-mouth learning. Games and Economic Behavior, 46:1–22, 2004.

[10] T. Bartz-Beielstein, K.E. Parsopoulos, and M.N. Vrahatis. Design and Analysis of Optimization Algorithms Using Computational Statistics. Applied Numerical Analysis & Computational Mathematics, 1(2):413–433, 2004.

[11] R. Bellman. Dynamic Programming. Dover Publications, 1957.

[12] R. Bellman. Adaptive Control Processes. Princeton University Press, 1961.

[13] G. Beni and J. Wang. Swarm intelligence in cellular robotic systems. In NATO Advanced Workshop on Robotics and Biological Systems, June 1989.

[14] B.A. Berg. Markov chain Monte Carlo simulations and their statistical analysis: with web-based Fortran code. World Scientific, 2004.

[15] J.M. Bishop. Stochastic searching network. Proceedings of the 1st IEE Conference on Artificial Neural Networks, pages 329–331, 1989.

[16] J.M. Bishop and P. Torr. The stochastic search network. Proceedings of the 1st IEE Conference on Artificial Neural Networks, pages 370–387, 1992.

[17] D. Blackwell and L. Dubins. Merging of Opinions with Increasing Information. The Annals of Mathematical Statistics, 33(3):882–886, 1962.

[18] V.D. Blondel, J.M. Hendrickx, and J.N. Tsitsiklis. On krause's consensus formation model with state-dependent connectivity. CoRR, abs/0807.2028, 2008. informal publication.

[19] E. Bonabeau, M. Dorigo, and G. Theraulaz. Swarm Intelligence: From Natural to Artificial Systems. Santa Fe Institute Studies in the Sciences of Complexity. Oxford University Press, USA, 1st edition, 1999.

[20] L. Breiman, J. Friedman, C.J. Stone, and R.A. Olshen. Classification and Regression Trees. Chapman and Hall/CRC, 1st edition, January 1984.

[21] H.J. Bremermann. Optimization through evolution and recombination. In M.C. Yovits, G.T. Jacobi, and G.D. Goldstein, editors, Proceedings of The Conference on Self-Organizing Systems, Chicago, Illinois, 1962. Spartan Books.

[22] X. Chen and Y.M. Li. A modified PSO structure resulting in high exploration ability with convergence guaranteed. IEEE Transactions on Systems Man and Cybernetics Part Bcybernetics, 37(5):1271–1289, 2007.

[23] M. Clerc and J. Kennedy. The particle swarm - explosion, stability, and convergence in a multidimensional complex space. Evolutionary Computation, IEEE Transactions on, 6(1):58–73, 2002.

[24] G. Como and F. Fagnani. Scaling limits for continuous opinion dynamics systems. In Proceedings of the 47th annual Allerton conference on Communication, control, and computing, pages 1562–1566, Piscataway, NJ, USA, 2009. IEEE Press.

[25] A. Czarn, C. MacNish, K. Vijayan, B.A. Turlach, and R. Gupta. Statistical exploratory analysis of genetic algorithms. IEEE Trans. Evolutionary Computation, 8(4):405–421, 2004.

[26] Ch. Darwin. On the Origin of Species by Means of Natural Selection. Murray, London, 1859.

[27] R. Dawkins. The Selfish Gene. Oxford University Press, September 1990.

[28] K. Deb. Multi-Objective Optimization Using Evolutionary Algorithms. John Wiley & Sons, Inc., New York, NY, USA, 2001.

[29] K. Deb and R.B. Agrawal. Simulated Binary Crossover for Continuous Search Space. Technical report, Departement of Mechanical Enginering, Indian Institute of Technology, Kanpur, India,1994.

[30] G. Deffuant, D. Neau, F. Amblard, and G. Weisbuch. Mixing beliefs among interacting agents. Adv. Complex Syst., 3(1–4):87–98, 2000.

[31] M. Doege. Rosenbrock function. http://en.wikipedia.org/wiki/File:Rosenbrock_function.svg. [Online; accessed 11-August-2011].

[32] M. Doege. The Himmelblau Function. http://en.wikipedia.org/wiki/File:Himmelblau_function.svg. [Online; accessed 11-August-2011].

[33] M. Dorigo. Optimization, Learning and Natural Algorithms. PhD thesis, Politecnico di Milano, Italy, 1992.

[34] M. Dorigo and T. St¨utzle. Ant colony optimization. Bradford Books. MIT Press, 2004.

[35] L.J. Eshelman and J.D. Schaffer. Real-coded genetic algorithms and interval-schemata. In L.D. Whitley, editor, FOGA, pages 187–202. Morgan Kaufmann, 1992.

[36] R. M. Friedberg. A learning machine: Part I. IBM Journal of Research and Development, 2(1):2–13, 1958.

[37] D. Gale and S. Kariv. Bayesian learning in social networks. Games and Economic Behavior, 45(2):329–346, November 2003.

[38] Y. Gao and Y. Duan. An adaptive particle swarm optimization algorithm with new random inertia weight. In D.S. Huang, L. Heutte, and M. Loog, editors, ICIC (3), volume 2 of Communications in Computer and Information Science, pages 342–350. Springer, 2007.

[39] J.E. Gentle, W. Härdle, and Y. Mori, editors. Handbook of Computational Statistics. Springer, 2004.

[40] D.E. Goldberg. Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1989.

[41] B. Golub and M.O. Jackson. Naive Learning in Social Networks: Convergence, Influence, and the Wisdom of Crowds. Preprint, 2007.

[42] E.R. Hansen and G.W. Walster. Global optimization using interval analysis. [Pure and applied mathematics]. Marcel Dekker, 2004.

[43] F. Herrera, M. Lozano, and J.L. Verdegay. Tackling real-coded genetic algorithms: Operators and tools for behavioural analysis. Artif. Intell. Rev., 12:265–319, August 1998.

[44] J.H. Holland. Outline for a logical theory of adaptive systems. J. ACM, 9:297–314, 1962.

[45] J.H. Holland. Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor, MI, USA, 1975.

[46] Ming J., Yupin L., and Shiyuan Y. Stagnation analysis in particle swarm optimization. In Swarm Intelligence Symposium, 2007. SIS 2007. IEEE, pages 92–99, april 2007.

[47] K.A. De Jong. An analysis of the behavior of a class of genetic adaptive systems. PhD thesis, University of Michigan, Ann Arbor, MI, USA, 1975.

[48] D. Karaboga. An idea based on Honey Bee Swarm for Numerical Optimization. Technical Report TR06, Erciyes University, October 2005.

[49] J. Kennedy. The particle swarm: social adaptation of knowledge. Proceedings of 1997 IEEE International Conference on Evolutionary Computation ICEC 97, pages 303–308, 1997.

[50] J. Kennedy. Population structure and particle swarm performance. In: Proceedings of the Congress on Evolutionary Computation (CEC 2002), pages 1671–1676. IEEE Press, 2002.

[51] J. Kennedy and R.C. Eberhart. Particle swarm optimization. In Neural Networks, 1995. Proceedings., IEEE International Conference on, volume 4, pages 1942–1948, August 2002.

[52] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi. Optimization by simulated annealing. Science, 220:671–680, 1983.

[53] U. Krause. A discrete nonlinear and non-autonomous model of consensus formation. In S. Elyadi, G. Ladas, J. Popenda, and J. Rakowski, editors, Communications in Difference Equations, pages 227–236. Gordon and Breach Pub., Amsterdam, 2000.

[54] M. Kronfeld, H. Planatscher, and A. Zell. The EvA2 optimization framework. In C. Blum and R. Battiti, editors, Learning and Intelligent Optimization Conference, Special Session on Software for Optimization (LION-SWOP), number 6073 in Lecture Notes in Computer Science, LNCS, pages 247–250, Venice, Italy, January 2010. Springer Verlag.

[55] F. Kursawe. A variant of evolution strategies for vector optimization. In Proceedings of the 1st Workshop on Parallel Problem Solving from Nature, PPSN I, pages 193–197, London, UK, 1991. Springer-Verlag.

[56] A. Lihu. O abordare eusocială în optimizarea stigmergică (Raport anual de cercetare nr. 1). http://www.yudu.com/item/details/90337/O-abordare-eusocial-----n-optimizarea-stigmergic--, September 2009. [Online; accessed 11-August-2011].

[57] A. Lihu. Raport anual de cercetare nr. 2. http://www.scribd.com/doc/77443904, September 2010. [Online; accessed 11-August-2011].

[58] A. Lihu and Ș. Holban. Particle swarm optimization with disagreements. In Y. Tan, Y. Shi, Y. Chai, and G. Wang, editors, ICSI (1), volume 6728 of Lecture Notes in Computer Science, pages 46–55. Springer, 2011.

[59] A. Lihu and Ș. Holban. Top five most promising algorithms in scheduling. In Proceedings of the 5th International Symposium on Applied Computational Intelligence and Informatics, pages 397–404, Timișoara, Romania, May 2009. IEEE.

[60] A. Lihu and Ș. Holban. Particle swarm optimization with disagreements on stagnation. In Radoslaw K., T.F. Chiu, C.F. Hong, and N. Nguyen, editors, Semantic Methods for Knowledge Management and Communication, volume 381 of Studies in Computational Intelligence, pages 103–113. Springer, 2011.

[61] A. Lihu and Ș. Holban. Real-valued genetic algorithms with disagreements. In D. Pelta, N. Krasnogor, D. Dumitrescu, C. Chira, and R. Lung, editors, NICSO, volume 387 of Studies in Computational Intelligence, pages 333–346. Springer, 2011.

[62] A. Lihu and Ș. Holban. A study on the minimal number of particles for a simplified particle swarm optimization algorithm. In Proceedings of the 6th IEEE International Symposium on Applied Computational Intelligence and Informatics (SACI), pages 299–303, Timi¸soara, Romania, May 2011. IEEE.

[63] J. Lorenz. A stabilization theorem for dynamics of continuous opinions. Physica A: Statistical Mechanics and its Applications, 355(1):217–223, September 2005.

[64] C. MacNish. Towards unbiased benchmarking of evolutionary and hybrid algorithms for real-valued optimisation. Connect. Sci, 19:361–385, December 2007.

[65] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller. Equation of state calculations by fast computing machines. Journal of Chemical Physics, 21, 1953.

[66] M.M. Millonas. Swarms, phase transitions and collective intelligence. In C. Langton, editor, Artificial Life III. Addison-Wesley, 1994.

[67] D.C. Montgomery. Design and Analysis of Experiments. John Wiley & Sons, 2006.

[68] P. Moscato. On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. Technical Report C3P Report 826, California Institute of Technology, 1989.

[69] H. Mühlenbein and D. Schlierkamp-Voosen. Predictive models for the breeder genetic algorithm i. continuous parameter optimization. Evol. Comput., 1:25–49, March 1993.

[70] J.A. Nelder and R. Mead. A Simplex Method for Function Minimization. The Computer Journal, 7(4):308–313, January 1965.

[71] A. Neumaier. Interval methods for systems of equations. [Encyclopedia of mathematics and its applications]. Cambridge University Press, 1990.

[72] E. Ozcan and C.K. Mohan. Particle swarm optimization: Surfing the waves. In Proceedings of the Congress on Evolutionary Computation, pages 6–9. IEEE Press, 1999.

[73] K.E. Parsopoulos and M.N. Vrahatis. Unified particle swarm optimization in dynamic environments. In Lecture Notes in Computer Science 3449, pages 590–599. Springer Verlag, 2005.

[74] K.E. Parsopoulos and M.N. Vrahatis. Particle Swarm Optimization and Intelligence: Advances and Applications. Premier Reference Source. Information Science Reference, 2010.

[75] M.E.H. Pedersen. Tuning and Simplifying Heuristical Optimization. PhD thesis, University of Southampton, UK, 2010.

[76] M.E.H. Pedersen and A.J. Chipperfield. Simplifying particle swarm optimization. Appl. Soft Comput., 10:618–628, March 2010.

[77] Y.G. Petalas, K.E. Parsopoulos, and M.N. Vrahatis. Entropy-based memetic particle swarm optimization for computing periodic orbits of nonlinear mappings. In IEEE Congress on Evolutionary Computation, pages 2040–2047. IEEE, 2007.

[78] Y.G. Petalas, K.E. Parsopoulos, and M.N. Vrahatis. Memetic particle swarm optimization. Annals OR, 156(1):99–127, 2007.

[79] F.M. De Rainville, F. A. Fortin, C. Gagné, M. Parizeau, and M. Gardner. Distributed Evolutionary Algorithms in Python. http://code.google.com/p/deap. [Online; accessed 11-August 2011].

[80] J. Renze and E.W. Weisstein. Extreme value theorem. from mathworld–a wolfram web resource. http://mathworld.wolfram.com/ExtremeValueTheorem.html. [Online; accessed 11 August-2011].

[81] C.W. Reynolds. Flocks, herds and schools: A distributed behavioral model. SIGGRAPH Comput. Graph., 21:25–34, August 1987.

[82] H.H. Rosenbrock. An automatic method for finding the greatest or least value of a function. The Computer Journal, 3(3):175–184, March 1960.

[83] J. Sacks, W.J. Welch, T.J. Mitchell, and H.P. Wynn. Design and analysis of computer experiments. Statistical science, 4(4):409–423, 1989.

[84] T.J. Santner, B.J. Williams, and W.I. Notz. The Design and Analysis of Computer Experiments. Springer Verlag, New York, 2003.

[85] L.J. Savage. The foundations of statistics. Dover Publications, 1972.

[86] Y. Shi and R.C. Eberhart. A Modified Particle Swarm Optimizer. In Proceedings of IEEE International Conference on Evolutionary Computation, pages 69–73, Washington, DC, USA, May 1998. IEEE Computer Society.

[87] Y. Shi and R.C. Eberhart. Parameter selection in particle swarm optimization. In V.W. Porto, N. Saravanan, D.E. Waagen, and A.E. Eiben, editors, Evolutionary Programming, volume 1447 of Lecture Notes in Computer Science, pages 591–600. Springer, 1998.

[88] Y. Shi and R.C. Eberhart. Empirical study of particle swarm optimization. In Proceedings of the 1999 Congress on Evolutionary Computation, volume 3, page 1950, 1999.

[89] Y. Shi and R.C. Eberhart. Fuzzy adaptive particle swarm optimization. In Proceedings of the 2001 Congress on Evolutionary Computation, volume 1, pages 101–106, 2001.

[90] L. Smith and P. Sorensen. Pathological outcomes of observational learning. Econometrica, 68(2):371–398, March 2000.

[91] R. Storn and K. Price. Differential evolution — a simple and efficient heuristic for global optimization over continuous spaces. J. of Global Optimization, 11:341–359, December 1997.

[92] P.N. Suganthan, N. Hansen, J.J. Liang, K. Deb, Y.P. Chen, A. Auger, and S. Tiwari. Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization. Technical report, Nanyang Technological University, Singapore, May 2005.

[93] A. Swanepoel. Alternative GnuPlot render sizes, Rosenbrock Function. http://twiki.org/cgi-bin/view/Plugins/GnuPlotPlugin. [Online; accessed 11-August-2011].

[94] K. Tang, X. Li, P.N. Suganthan, Z. Yang, and T. Weise. Benchmark functions for the CEC 2010 special session and competition on Large-Scale global optimization. Technical report, University of Science and Technology of China, November 2009.

[95] A. Törn and A. Zilinskas. Global optimization. Springer-Verlag New York, Inc., New York, NY, USA, 1989.

[96] V. Černý. Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. Journal of Optimization Theory and Applications, 45(1):41–51, 1985.

[97] F. Van Den Bergh. An analysis of particle swarm optimizers. PhD thesis, University of Pretoria, Pretoria, South Africa, 2002.

[98] D.H. Wolpert and W.G. Macready. No free lunch theorems for search. Tech. Rep. No. SFI-TR 95-02-010, Santa Fe Institute, NM, 1995.

[99] D.H.Wolpert andW.G. Macready. No free lunch theorems for optimization. IEEE Trans. Evolutionary Computation, 1(1):67–82, 1997.

[100] D.H. Wolpert and W.G. Macready. Coevolutionary free lunches. IEEE Trans. Evolutionary Computation, 9(6):721–735, 2005.

[101] C. Worasucheep. A particle swarm optimization with stagnation detection and dispersion. In IEEE Congress on Evolutionary Computation, pages 424–429. IEEE, 2008.

[102] X.F. Xie, W.J. Zhang, and Z.L. Yang. Dissipative particle swarm optimization. In Proceedings of the 2002 Congress on Evolutionary Computation, volume 2, pages 1456–1461, Washington, DC, USA, 2002. IEEE Computer Society.

[103] J.H. Zwiebel, D. Vayanos, and P.M. DeMarzo. Persuasion bias, social influence, and uni-dimensional opinions. Research Papers 1719, Stanford University, Graduate School of Business, November 2001.