FEDERAL UNIVERSITY OF PARÁ
INSTITUTE OF EXACT AND NATURAL SCIENCE
GRADUATE PROGRAM IN COMPUTER SCIENCE

Reginaldo Cordeiro dos Santos Filho

Global Continuous Optimization with Particle Swarm Enhancements

Belém-PA

2019

Reginaldo Cordeiro dos Santos Filho

Global Continuous Optimization with Particle Swarm Enhancements

Thesis submitted to the Graduate Program in Computer Science of Federal University of Pará in partial fulfillment of the requirements for the Degree of Doctor of Philosophy in Computer Science, Area of Concentration in Computer Systems, Research Line in Computational Intelligence.

Advisor: Prof. Dr. Claudomiro de Souza de Sales Júnior

Belém-PA

2019

Dados Internacionais de Catalogação na Publicação (CIP) de acordo com ISBD
Sistema de Bibliotecas da Universidade Federal do Pará
Gerada automaticamente pelo módulo Ficat, mediante os dados fornecidos pelo(a) autor(a)

S237g Santos Filho, Reginaldo Cordeiro dos.
Global Continuous Optimization with Particle Swarm Enhancements / Reginaldo Cordeiro dos Santos Filho, . — 2019.
143 f. : il. color.

Orientador(a): Prof. Dr. Claudomiro de Souza de Sales Júnior
Tese (Doutorado) - Programa de Pós-Graduação em Ciência da Computação, Instituto de Ciências Exatas e Naturais, Universidade Federal do Pará, Belém, 2019.

1. Global Continuous Optimization. 2. Particle Swarm Optimization. 3. Rotation Variance. 4. Deterministic Algorithm. 5. Metaheuristic Approach. I. Título.

CDD 006.3

UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE CIÊNCIAS EXATAS E NATURAIS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

REGINALDO CORDEIRO DOS SANTOS FILHO

OTIMIZAÇÃO CONTÍNUA GLOBAL COM MELHORAMENTOS PARA O ENXAME DE PARTÍCULAS

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal do Pará como requisito para obtenção do título de Doutor em Ciência da Computação, defendida e aprovada em 18/02/2019, pela banca examinadora constituída pelos seguintes membros:

Prof. Dr. Claudomiro de Souza de Sales Junior
Orientador – PPGCC/UFPA

Prof. Dr. Filipe de Oliveira Saraiva
Membro Interno – PPGCC/UFPA

Prof. Dr. Josivaldo de Souza Araújo
Membro Interno – PPGCC/UFPA

Prof. Dr. Ronnie Cley de Oliveira Alves
Membro Interno – PPGCC/UFPA

Prof. Dr. Roberto Célio Limão de Oliveira
Membro Externo – UFPA

Prof. Dr. Adam Dreyton Ferreira dos Santos
Membro Externo – UNIFESSPA

Visto:
Prof. Dr. Bianchi Serique Meiguins
Coordenador do PPGCC/UFPA

I would like to dedicate this thesis to my mother.

Acknowledgements

This thesis would not be possible without the help and support of kind people.

I especially thank my beloved mom, my dear aunt and my fantastic grandmother. My hard-working family have sacrificed their lives for me and provided unconditional love and care. I do believe that I would not have made it this far without them. I know I always have my family to count on when times are tough.

I would not have contemplated this road if not for my lovely wife, who instilled love in all senses within me. She has been the best of friends along this journey. Thanks my dearest for being patient with me, for being beside me when no one could be, and for rekindling my dreams when no one believed on me.

I would like to express my heartfelt gratitude to Professor Claudomiro de Souza de Sales Júnior, my PhD advisor, who I have enjoyed the opportunity to learn from not only how to perform research in a scientific era, but also how to think out-of-the-box. I could not be prouder of my academic roots and hope that I can in turn pass on the research values and the dreams that he has given to me.

I would like to thank my friend, Full Professor PhD João Crisóstomo Weyl Albuquerque Costa, who was truly supportive during the doctorate period. He has always cared about me and my future and provided great insights for me to pick my path in the best direction. My sincere thanks to my friend and Professor PhD Gilvan Soares Borges, who has been teaching me mathematical tools to deal with difficult formulations in the field of complex systems. I am so proud of working with you, thank you both.

My special *thanks* goes to my friends of Applied Electromagnetism Laboratory (LEA) who have supported me over the years. Thank you very much for all memorable moments you shared with me. I also have enjoyed many useful and entertaining discussions with my personal friends in my life who I admire and I owe part of all my experience.

I would also like to thank all the people who listened patiently when I discussed some of my ideas with them, for their feedback and insight.

This work was funded by Coordination for the Improvement of Higher Education Personnel (CAPES) – a Brazilian Federal Agency for Support and Evaluation of Graduate Education within the Ministry of Education of Brazil.

"Man is condemned to be free."
Jean-Paul Charles Aymard Sartre

Abstract

Numerous optimization algorithms have been proposed to solve computing and engineering problems involving optimization of a set of real parameters. A global continuous optimization problem is generally NP-hard which poses a difficult and complex task to find the true global optimum solution. In this context, popular metaheuristic approaches based on a population of simple structures have emerged during the recent years. One of them is the well-known Particle Swarm Optimization (PSO) algorithm which is a non-deterministic, randomized, nature-inspired metaheuristic specialized in solving continuous black-box optimization problems. PSO has been used by many experts of different domains of science, as the algorithm has the potential to solve complex problems with simple mathematical formulations. Although this method is widely used in real-world applications, there are also intrinsic drawbacks embedded in the algorithm's structure. Some of them are known to all, such as the search mechanism wastes computational efforts due to random walks, the convergence process is usually slow, some points are revisited during the search process, prominent areas are not properly investigated, the algorithm is prone to premature convergence and the algorithm's parameters are problem-dependent. In this thesis, the author presents enhancements to the original PSO by incorporating deterministic characteristics and rotation matrices. A semi-autonomous particle swarm optimizer, termed SAPSO, which uses gradient-based information and diversity control, is provided. This work also investigates the influence of rotation and information exchange on the performance of PSO. Four PSO versions which include the presence or absence of rotation variance, and the fast or late information exchange among particles are investigated. As final outcome of this thesis, an algorithm, coined as Invariant SAPSO (ISAPSO), is reported to improve the capability of SAPSO algorithm by embedding new search mechanisms. The numerical simulations revealed statistical significant results when ISAPSO is evaluated on benchmark optimization problems and compared to other related PSO-based algorithms. No statistically significant results were observed between rotation variance or invariance, whereas the fast information exchange outperformed the late information exchange. In addition, a rigorous methodology based on a reliable number of executions and statistical hypothesis tests is conducted to strengthen the discussions.

Key-words: Global Continuous Optimization, Particle Swarm Optimization, Rotation Variance, Deterministic Algorithm, Metaheuristic Approach.

Resumo

Muitos algoritmos de otimização têm sido propostos para resolver problemas da engenharia e da computação que envolvem a otimização de um conjunto de parâmetros reais. Um problema de otimização contínua é geralmente classificado como NP-difícil, o que o torna uma tarefa complexa de encontrar a solução ótima global. Neste contexto, abordagens populares de metaheurísticas baseadas em população de estruturas simples têm surgido durante os últimos anos. Uma delas é o algoritmo de Otimização por Enxame de Partículas (PSO) que é não determinístico, randômico, inspirado na natureza e especializado em resolver problemas de otimização "caixa-preta". PSO tem sido usado por muitos especialistas de diferentes áreas da ciência, uma vez que o algoritmo tem potencial para resolver problemas complexos por meio de formulações matemáticas simples. Embora este método seja amplamente utilizado em aplicações reais, existem desvantagens intrínsecas embarcadas na estrutura do algoritmo. Algumas delas são bem conhecidas, a citar o mecanismo de busca desperdiça tempo computacional devido ao *random walk*, o processo de convergência é geralmente lento, alguns pontos são revisitos durante o processo de busca, áreas promissoras não são apropriadamente investigadas, o algoritmo está propenso a convergência prematura e os parâmetros do algoritmo são dependentes do problema. Nesta tese, o autor apresenta melhorias para o PSO original através da incorporação de características determinísticas e matrizes de rotação, além de realizar análises empíricas sobre rotação e troca de informação entre as partículas. Um otimizador por enxame de partículas semi-autônomas, cunhado de SAPSO, que usa informações de gradiente e controle de diversidade é fornecido. Esta tese também investiga a influência da rotação e a troca de informação entre partículas no desempenho do PSO. Quatro versões do PSO que incluem a presença e ausência da propriedade *rotation variance*, além da troca rápida e lenta de informações entre partículas são investigados. Como produto final desta tese, um algoritmo cunhado de ISAPSO é desenvolvido com o objetivo de fortalecer a capacidade do algoritmo SAPSO incorporando novos mecanismos de busca. As simulações numéricas revelaram resultados convincentes quando o ISAPSO é avaliado em um conjunto de problemas de otimização e comparados com outros algoritmos PSO. Não foram observados resultados estatisticamente significantes na comparação entre as propriedades *rotation variance* ou *invariance*, enquanto que a troca rápida de informação superou a troca lenta de informação entre as partículas. Além disso, uma metodologia rigorosa baseado em um número confiável de execuções e testes de hipóteses estatísticas é conduzida para fortalecer as discussões.

Palavras-chave: Otimização Global Contínua, Otimização por Enxame de Partículas, Variação de Rotação, Algoritmo Determinístico, Abordagem Metaheurística.

List of Illustrations

List of Tables

List of abbreviations and acronyms

UFPA	Universidade Federal do Pará
PPGCC	Programa de Pós-Graduação em Ciência da Computação
PSO	Particle swarm optimization
SAPSO	Semi-Autonomous Particle Swarm Optimizer
ISAPSO	Invariant Semi-Autonomous Particle Swarm Optimizer
APSO	Adaptive Particle Swarm Optimization
ARPSO	Diversity-guided attractive and repulsive PSO
GPSO	Gradient-based PSO
HGPSO	Hybrid gradient descent PSO
SPSO	Standard Particle Swarm Optimization
WPSO	Wilke Particle Swarm Optimization
BPSO	Bonyadi Particle Swarm Optimization
PSO-TVIW	PSO with Time-Varying Inertia Weight
DGHPSOGS	Diversity-guided hybrid PSO based on gradient search
GA	Genetic algorithm
BFGS	Broyden-Fletcher-Goldfarb-Shanno algorithm
CEC	Congress on Evolutionary Computation
DNPP	Distribution of all Next Possible Position

Contents

1 Introduction

Stochastic search methods, such as Particle swarm optimization (PSO) and Genetic algorithm (GA), are well-known population-based metaheuristics to perform global search (KENNEDY; EBERHART, 1995; GOLDBERG, 1989). These algorithms have been used throughout the years as multidisciplinary methods to face convex, non-convex, unimodal, multimodal, unidimensional and multidimensional optimization problems. The practitioners resort to stochastic search method when deterministic algorithms are not suitable to deal with the optimization problem. Most of the computing and engineering optimization problems are NP-hard, and for this reason the scholars must check the whole continuous search space to ensure that a result is not suboptimal, and this task obviously would lead to a search with prohibitive time. In this scenario, looking for a solution at least near to the global optimum is plausible. This is where the stochastic search methods are most suitable.

The literature of PSO is eclectic in terms of applying the metaheuristic in different domains, starting with toy- or puzzle-like problems, such as n-queen (AHMED et al., 2012; WANG; LIN; YANG, 2012), sudoku (MORAGLIO; TOGELIUS, 2007; HEREFORD; GERLACH, 2008), chess (DURO; OLIVEIRA, 2008); passing through industries-like problems, such as processing of ore dressing plant (HUANG et al., 2007), long-term production scheduling problem of the open pit mines (KHAN; NIEMANN-DELIUS, 2015), structural health monitoring applied in the assessment of bridges (SANTOS et al., 2016); also, as tool for training a machine learning algorithm, such as feed-forward neural networks (VILOVIĆ; BURUM; MILIĆ, 2009; CH; MATHUR, 2012; AKSU; COBAN, 2013) or hybridizes with other evolutionary algorithms like GAs (JUANG, 2004; BENTO et al., 2013; XU et al., 2015), ant-colony (DUAN; YING, 2009), differential evolution (HAO; GUO; HUANG, 2007), among others. This vast application domain of PSO is present in the literature of global optimization mainly because of its simplicity regarding its development and fundamental formulation, addressing two types of application domains: discrete binary (KENNEDY, 1997a) and continuous codifications (KENNEDY; EBERHART, 1995; EBERHART; KENNEDY, 1995). Therefore, pushing forward the frontier of the PSO knowledge can contribute to the area of swarm intelligence.

Such algorithm's popularity has brought much attention to the theoretical issues about the fundamental operation of PSO. To name a few examples: Ozcan and Mohan analysed a simplified particle system to determine the trajectory of an isolated particle in one dimension (OZCAN; MOHAN, 1998) and multidimensional search space (OZCAN et al., 1999); Clerc and Kennedy provided an algebraic and analytical view of particle's trajectory to understand how the swarm searches the problem space (CLERC; KENNEDY,

2002); Van Den Bergh and Engelbrecht investigated particle's trajectory for general swarms, including inertia term (BERGH; ENGELBRECHT, 2006), and they also provided a formal mathematical proof that swarm converges to a stable point; Janson and Middendorf showed that particles have a clear bias in their movement direction which depends on the direction of the coordinate axes (JANSON; MIDDENDORF, 2007); Poli introduced a method to exactly determine the moments of a PSO sampling distribution and explained how they change over any number of iterations (POLI, 2009); Spears *et al.* showed that the rotation variance is related to the concentration of particles along lines parallel to the coordinate axes (SPEARS; GREEN; SPEARS, 2010); Schmitt and Wanka pointed out possible reasons related to the swarm stagnation at non-optimal points and the particles predilection of walking parallel to the axes (SCHMITT; WANKA, 2013b; SCHMITT; WANKA, 2013a). Although great strides have been discussed throughout the past years, there are still enhancements to propose in this field of study, such as increasing the search potential of PSO with deterministic steps and also providing mathematical structures to understand crucial behaviors of particles during the search process.

The deterministic optimization algorithms far outweigh the non-deterministic ones on unimodal functions. However, classical algorithms, such as gradient descent and Newton's method, are strongly dependent on the quality of the initial guess and easily get trapped into local optima of multimodal functions (HORST; TUY, 1996). On the contrary, non-deterministic optimization methods, such as PSO and GAs perform global optimization, however they waste computational time wandering the search space as a result of the random walk influence. This phenomenon also contributes significantly to the exploration of areas already visited, wasting computational efforts re-evaluating previously observed candidate solutions. To the best of the author's knowledge, there are no stochastic algorithms with finite memory that store the points from the continuous search space already evaluated during the search process[1]. Furthermore, there are no embedded mechanisms of exploiting specific areas of the search space efficiently during the early iterations. Often, the convergence process is slow, some points are revisited during the search process, a prominent area of the search space is not properly investigated by the random walk mechanism, and the algorithms are prone to premature convergence, specially when they are highly dependent of their initial parameters.

The thesis provides enhancements to the PSO-based algorithms, covering some aforementioned challenges. The main contributions of this thesis are highlighted in three parts: a new PSO version called SAPSO, an empirical analysis of classical PSO versions, and an improved version of SAPSO algorithm coined as ISAPSO, which is strictly rotationally invariant under rotation of the coordinate system. The next section provides a description about all contributions found in this thesis in a chronological order.

[1] A modification in the Tabu search methodology (GLOVER, 1989; GLOVER, 1990) to work properly in a continuous search space may handle this feature.

1.1 Thesis contributions

A brief description of each aforementioned contribution is given in this section. Section 1.1.1 is related to foundations of a new PSO version, termed as SAPSO, which brings deterministic characteristics to the search process of PSO. The algorithm also embeds a mechanism of attraction and repulsion of particles according to the diversity value of the swarm. Section 1.1.2 outlines about the study of four classical PSO versions on a set of multimodal optimization problems. The main focus of this analysis is to investigate whether rotation variance and information exchange among particles affect the performance of PSO-based algorithms. Section 1.1.3 describes a new version of SAPSO algorithm, coined as Invariant SAPSO (ISAPSO), which reduces the computational time to run the algorithm, removes heuristics such as bound handling and velocity clamps, and embeds a rotation matrix to introduce a perturbation in the movement of particles.

1.1.1 A semi-autonomous particle swarm optimizer

The first contribution of the thesis is the SAPSO algorithm which uses gradient-based information and diversity control to optimize unimodal and multimodal functions. The main idea behind SAPSO algorithm is to reduce computational efforts of local investigation and to provide a mechanism to escape from local optima. The term *autonomous* means a decision that each particle must take at each iteration of the algorithm. Each particle must decide between exploit its current neighborhood through the gradient information or to follow in the direction of the current global best position. Moreover, the term *semi* places the entire population under a general rule of attraction and repulsion which is based on the diversity value of the swarm. Therefore, the term *semi-autonomous* characterizes the particles of the SAPSO algorithm. With this embryonic idea, at least two of the main optimization challenges are faced in the roots of the proposed algorithm: 1) to properly investigate prominent areas of the search space and 2) to avoid premature convergence. This model of global optimization promises an upgrade on the current metaheuristic performances.

SAPSO algorithm is further compared to a diversity-guided attractive and repulsive PSO (ARPSO) (VESTERSTRØM; RIGET, 2002), gradient-based PSO (GPSO) (NOEL, 2012), and a diversity-guided hybrid PSO based on gradient search (DGHPSOGS) (HAN; LIU, 2014). The ARPSO algorithm introduced the idea of diversity control along the iterations, while the GPSO algorithm was the first to consider the gradient information with PSO, and DGHPSOGS applied both gradient and diversity control information in a hybrid approach. When the ideas of those algorithms are used together, they come up with advanced results in the optimization field. The performances of the algorithms are evaluated on a suite of test functions based on the De Jong's benchmark optimization problems.

1.1.2 An empirical analysis of classical PSO versions

The empirical analyses are referred to as four PSO-based approaches: rotation variance, rotation invariance, late information exchange, and fast information exchange. The first two versions are related to the random variables handled by the velocity update equation of particles. Rotation variance embeds two vectors of random numbers in the cognitive and social components, while rotation invariance embodies scalar random variables. The practical difference is about how particles move toward the cognitive and social memories in the search space. In (WILKE; KOK; GROENWOLD, 2007a; WILKE; KOK; GROENWOLD, 2007b), the authors show that the directional diversity is evident in the rotation variant PSO version, whereas the rotation invariance reduces the search distribution of particles, which in turn aggravates the probability of premature convergence. Both rotational versions are still present in the literature, where the variant version can be seen in (CHEN et al., 2017; LYNN; ALI; SUGANTHAN, 2018; CHEN et al., 2018; LYNN; SUGANTHAN, 2017; ZHANG et al., 2017; YU; WANG; WANG, 2016) and the invariant one in (ESPITIA; SOFRONY, 2018; XU; YU, 2018; TIAN; SHI, 2018; LI; CHENG, 2017; GOU et al., 2017; CHEN et al., 2018). As far as the author know, the literature provides no consistent distinction of when to apply one version or another.

Beyond two rotational versions of PSO algorithm, another important discussion is about the fast and late information exchange mechanisms. Both options change how information exchange is spread through the swarm. The former forwards the information about the best particle's position right after a new position that deserves exploitation is discovered. Some examples of this implementation can be found in (MARINI; WALCZAK, 2015; BEHESHTI; SHAMSUDDIN, 2015; NOEL, 2012; ZHAN et al., 2011; TEWOLDE; HANNA; HASKELL, 2009; LIANG et al., 2006). The latter postpones the information of promising areas and may skip intermediate local optima discovered by isolated particles in the same iteration, consequently leaving a more selective position for the end of the iteration. Some works with this approach are found in (ISSA et al., 2018; KUMARI et al., 2017; GBENGA; RAMLAN, 2016; JENSI; JIJI, 2016; BEHESHTI; SHAMSUDDIN; HASAN, 2013; ROBATI et al., 2012). A misinterpretation of the classical PSO algorithm might be the reason for such differentiation, as this question is not clear enough in the original papers (EBERHART; KENNEDY, 1995; KENNEDY; EBERHART, 1995; KENNEDY, 1997a; SHI; EBERHART, 1998b; SHI; EBERHART, 1998a; SHI; EBERHART, 1999).

Although there are differences, both approaches work satisfactorily. However, no consistent justification is given about when or why to apply one instead of another. Following this idea, this thesis also evaluates the performance of both approaches along with variant and invariant issues on CEC 2017 benchmark functions. Besides, the methodology follows rigorous rules defined by Maurice Clerc (CLERC, 2012a), where different unbiased test functions, sufficient number of executions, different statistical measures, and two sta-

tistical hypothesis tests are considered. These principles aim to avoid the misinterpretation of results by reducing the possibility to obtain biased results by chance. Furthermore, a method to select a minimum number of algorithm's execution is also provided whose law of large numbers is the crucial motivation.

1.1.3 ISAPSO algorithm: a rotationally invariant PSO version

An improvement of SAPSO algorithm, which is named Invariant SAPSO (ISAPSO), is proposed by this thesis as a final outcome. ISAPSO algorithm arises from the foundations of its predecessor and also from the final outcomes on the empirical analyses of four PSO versions described previously (Section 1.1.2). ISAPSO algorithm enhances the directional diversity by introducing a rotation matrix in the velocity update equation. This rotation matrix plays a role when the swarm is in the repulsion phase due to the small perturbation in the directions of the social and gradient components. Therefore, the attraction and repulsion mechanism makes particles susceptible to move toward prominent areas of the search space through a different angle after each repulsion be activated. As a consequence, one can expect a larger search distribution for each particle and the avoidance of trapping into local minima.

Additionally, ISAPSO algorithm decreases the computational time by performing gradient calculus only when is required. Previously, SAPSO algorithm has performed partial derivatives every time a particle moves without considering its autonomous decision. In the new algorithm's version, first the particle takes the decision, and if the particle decides to follow the gradient information, then it performs gradient calculus. Besides that, all heuristics such as bound handling and velocity clamps were removed, which simplified the coding. Moreover, the main parameters, such as inertia weight, social and gradient coefficients were carefully chosen to be static and to serve as an initial guess for different optimization problems.

A consistent mathematical proof about the rotation invariance is formulated later where it is shown that the algorithm is strictly rotationally invariant in the attraction phase and rotationally invariant in a stochastic sense in the repulsion phase. ISAPSO algorithm is also evaluated on CEC 2017 benchmark functions of real parameters by considering different statistical measures. The methodology used by the experiments also follows the Clerc's rules. Furthermore, two statistical hypothesis tests are applied on the results of ISAPSO algorithm and other related PSO algorithms to evaluate the statistical significance of their performances.

1.2 Contextualization

Since the classical PSO algorithm was first published by Kennedy and Eberhart in 1995 (KENNEDY; EBERHART, 1995; EBERHART; KENNEDY, 1995), the field of swarm intelligence has been increasing in last years. Many improvements were proposed, tested, and compared in the literature through different perspectives by distinct practitioners. Although many applications with swarm intelligence are possible in current days, much work has been done to provide a mature metaheuristic able to be applied in such different domains of science.

In this contextualization, one can separate works considered important in history of PSO. Most of the proposed changes in the foundations of the algorithm are used up today. After the first published paper in 1995, a second work (KENNEDY, 1997b) presented a study about four models of knowledge: full, cognitive-only, social-only and selfless. The models are related to the velocity update equation. Some empirical simulations concluded that the social-only model (i.e., a version of the velocity update equation where only the inertial and social components are used) contributed more when solving simple optimization problems. Shi and Eberhart (SHI; EBERHART, 1998a) introduced a new parameter to the velocity update equation termed as *inertia weight*. Numerical simulations were performed to illustrate the impact of this parameter on the performance of PSO. The inertia weight plays an important role of controlling the exploration and exploitation phases. That paper was also the first one to bring the idea of linear or non-linear decreasing of inertia weight. In the next year, the same authors (SHI; EBERHART, 1999) formalized a linear decreasing inertia weight approach and they extensively evaluated the performance of PSO through experimental studies on four non-linear functions well-known in the literature. The experimental results illustrated that the PSO has the ability to quickly converge to a local optima at the end of a run due to the utilization of a linearly decreasing inertia weight. It was also observed that PSO may fail to find the required optima in cases where the problem to be solved is too complicated and complex. But to some extent, this can be overcome by employing a self-adapting strategy for adjusting the inertia weight. This root idea was later updated and investigated by other authors (RATNAWEERA; HALGAMUGE; WATSON, 2004; YAMAGUCHI; YASUDA, 2006; ZHAN et al., 2007; ZHAN et al., 2009).

Still in the 90s, Maurice Clerc (CLERC, 1999) reported for the first time a *constriction term* in the literature of PSO (later would be called *constriction factor*). This paper presented a purely deterministic algorithm called Swarm & Queen, with just one equation, one confidence coefficient, and one "memory" parameter. It was an attempt to build a deterministic adaptive PSO. Some non-convex functions were evaluated with this method, but a clearer investigation of the method was left for future work. Simultaneously, Eberhart and Shi (EBERHART; SHI, 2000) were evaluating through empirical simulation both

inertia weight and constriction factor in PSO. The authors stated that the PSO algorithm with the constriction factor can be considered as a special case of the algorithm with inertia weight since the three parameters (inertia weight, cognitive coefficient, and social coefficient) are connected through the principal equations derived from the constriction factor analysis. Finally in 2002, Clerc *et al.* analysed in (CLERC; KENNEDY, 2002) a particle's trajectory as it moves in discrete time (the algebraic view), then progresses to the view of it in continuous time (the analytical view). These analyses led to a generalized model of the algorithm, containing a set of coefficients to control the system's convergence tendencies. In other words, the application of constriction coefficients allows control over the dynamical characteristics of the particle swarm, including its exploration versus exploitation propensities.

One can affirm the discoveries of inertia weight, constriction factor, and different proposed approaches to deal with parameter tuning have improved the quota of optimization problems solved by PSO. Furthermore, the popularity of PSO, due to its simplicity and easy implementation, has capillarized and leveraged in many different areas of science. In recent years, most implementations are applications of the optimizer on popular context, such as load balancing in cloud computing (ACHARYA; MEHTA; SAINI, 2016), evolve internal parameters of deep learning processing layers (KHALIFA et al., 2017), load balancing and control in 5G heterogeneous networks (SHAMI; GRACE; BURR, 2018), localization of sensor nodes in internet of things (RAUNIYAR; ENGELSTAD; MOEN, 2018), path planning of multi-robots in unknown environments (THABIT; MOHADES, 2019), train a type-2 fuzzy neural networks with PSO to control quadcopters for an autonomous quality inspection over rice farms (CAMCI et al., 2018), and control the power of the sources based on an interlinking converter used to connect two micro grid systems in a renewable energy system (SAAD; EL-SATTAR; MANSOUR, 2018). As seen in this section, the presence of PSO algorithm in such distinct scenarios shows that providing new strategies to strengthen its search process can contributes to propagate this non-deterministic approach to an even more consolidated state of usage. In this context, this thesis is concerned about studying the foundations of PSO algorithm, extract important information on the algorithm's search premises, and ultimately yielding reliable improvements to the swarm intelligence area.

1.3 Related works

In this section, the principal works strictly related to the subjects discussed by this thesis are highlighted. The enhancements described by this thesis have narrow relations with the following topics: diversity control of the swarm, gradient-based information, properties of *rotation variance* and *rotation invariance*, directional diversity and rotation matrices. These topics have great relevance to the domain of PSO algorithms due to the features associated with them, which are discussed below.

Any population-based metaheuristic is susceptible of suffering from the effect of premature convergence, i.e., the elements constituting the population get stuck into a local optima during the early iterations. This happens when the elements erroneously believe that a promising area in the search space was discovered. In the PSO algorithm, the only information exchanged by the particles is the global best position, i.e., the best positioned particles with the best fitness value. This position guides the whole swarm toward prominent areas of the search space. It is worth pointing out that the default implementation of PSO algorithm has no mechanisms to detect a local optima or even to escape from these pitfalls. With this in mind, the diversity controlling mechanism equips the swarm with a valuable information about the crowding and dispersion of particles throughout iterations.

The diversity controlling approach was first proposed in (VESTERSTRØM; RIGET, 2002). This work presented a diversity-guided particle swarm optimizer, termed as ARPSO algorithm. ARPSO implements the idea of diversity control over the swarm, which is an important feedback extracted from the swarm along the iterations of the algorithm. The authors of this paper used this information to decide between two designs called *attraction* and *repulsion*. The former is a state where the particles follow the global tendency of the swarm, i.e., the global best particle. The latter is a state where the particles follow the opposite direction of the global tendency. Since the particles crowd in some region of the search space, which is possibly a local minimum, this mechanism of attraction and repulsion is desired as it can detect this situation and sooner can repel each particle from its current positions through the scheme of repulsion. The repulsion scheme will push the particles in the opposite direction of the global best position while the best solution found so far remains intact. The diversity control of the swarm is an important feedback given by the swarm and must be further investigated.

The introduction of gradient-based PSO algorithm, named as GPSO, is published in (NOEL, 2012). The root idea is to use gradient directions to accurate local exploitation around the global best position, as well as to strengthen the global exploration provided by the PSO. GPSO algorithm combines features of stochastic and deterministic optimization schemes and avoids their weaknesses. This approach allows an accurate final solution to be computed while retaining the ability to explore better solutions. Later in (HAN; LIU, 2014), another relevant paper presented both the diversity control and gradient-based information as strategies to interchange two PSO algorithms during the search process. The new approach is a diversity-guided hybrid PSO based on gradient search, termed as DGHPSOGS. In this hybrid PSO, the searching alternates between an adaptive PSO (classical PSO with dynamic inertia weight) and DGPSOGS according to the diversity value of the swarm. The adaptive PSO algorithm led the swarm to converge to local minima, while DGPSOGS algorithm kept the diversity of the swarm adaptively as well as searching in the negative gradient direction.

Although diversity control and gradient-based information are important strides to consider by this thesis, the following works are related to fundamental issues of classical PSO versions. The authors concern about the properties of *rotation variance* and how the dynamics of particle's motion deal with rotation of the coordinate systems. When an optimization algorithm satisfies the property *rotation variance*, it means its performance may deteriorate as the coordinate system is manipulated, i.e., the algorithm is sensitive to the rotation of the search space, whereas *rotation invariance* preserves its performance while the coordinate system is rotated. In (WILKE; KOK; GROENWOLD, 2007a), the authors investigated the significance of diversity in the PSO algorithm by studying two different implementations of PSO: rotatinally invariant (known as *linear* in the paper) and rotationally variant (called as *classical*) PSO formulations. The only difference between both versions is the velocity update equation. The first implies scalar random variables, while the second works with vectors of random variables. They showed that particle trajectories collapse to line searches in d-dimensional space when linear PSO velocity update equation is used. The classical formulation does not suffer this drawback. Instead, directional diversity stochastic search trajectories are retained, which in turn helps to alleviate premature convergence.

Wilke *et al.* also provided mathematical tools to proof whether a metaheuristic approach is invariant of the scale and frame (i.e., translation and rotation) in which an objective function is posed (WILKE; KOK; GROENWOLD, 2007b). They have found that linear velocity update equation is scale and frame invariant, but that the classical velocity update equation lacks rotational invariance. The property comes with two consequences: linear velocity update equation lacks diversity, resulting in particle trajectories that collapse to line searches, and in contrast, the classical velocity update equation maintains diverse particle trajectories. Additionally, the authors illustrated that diversity and invariance are not necessarily exclusive, they proposed a new velocity update equation based on rotation matrix, called Wilke PSO (WPSO) algorithm in this thesis. This update equation is rotationally invariant and at the same time directionally diverse. This is achieved through consistent perturbation of the search directions. In Spears *et al.* (SPEARS; GREEN; SPEARS, 2010), the authors provided a mathematical relation between the property *rotation variance* and a bias, i.e., a tendency that particles have to navigate in directions parallel to the coordinate axes. This phenomenon might be the reason why performance is changed by the rotation of the search space, but no consistent proof is given in that paper.

Another relevant work was described by Maurice Clerc in (CLERC, 2012b) where a new variation, named Standard PSO (SPSO), would be used as a baseline algorithm for comparisons. By the time a new PSO version is proposed, it should be compared with the SPSO. This PSO version changes the way a particle moves through the search space by developing a new velocity update equation based on a hyperspherical search distribution, i.e., hyperspherical distribution is formed by a center point (average position

of three points: particle's position, local memory of the particle, and best global memory of the swarm) and a radius formed by the distance between the center and the particle. A random point from a uniform distribution is drawn inside this hypersphere and then the particle moves to this position. In this way, particle is not biased by the coordinate system. Thus, SPSO version is rotationally invariant. Later in (ZAMBRANO-BIGIARINI; CLERC; ROJAS, 2013), the authors evaluated the standard PSO version on 28 test functions designed for the Special Session on Real-Parameter Single Objective Optimization at CEC-2013. Results showed an outstanding performance of SPSO-2011 for the family of unimodal and separable test functions with a fast convergence to the global optimum.

In (BONYADI; MICHALEWICZ; LI, 2014), the proposed PSO can be considered an improvement of WPSO algorithm described in (WILKE; KOK; GROENWOLD, 2007b). The authors examined several issues associated with the multiplication of personal and social influence vectors by such random matrices, including uncontrollable changes in the length and direction of these vectors, weak direction alternation for the vectors that are aligned closely to coordinate axes resulting in preventing the swarm from further improvement in some situations, and limitation in particle movement to one orthant resulting in premature convergence in some situations. They proposed the use of a randomly generated rotation matrices (rather than an approximated rotation matrix in WPSO) in the velocity updating equation of the particle swarm optimizer. This approach makes it possible to control the impact of the random components (i.e. the random matrices) on the direction and length of personal and social influence vectors separately. As a result, all the above mentioned issues were effectively addressed. They also proposed to use the Euclidean rotation matrices for rotation because it preserves the length of the vectors during rotation, which makes it easier to control the effects of the randomness on the direction and length of vectors.

Bonyadi *et al.* in (BONYADI; MICHALEWICZ, 2014) also developed a locally convergent rotationally invariant PSO algorithm. The motivations are related to the avoidance of stagnation of particles in some points in the search space, inability to change the value of one or more decision variables, poor performance when the swarm size is small, lack of guarantee to converge even to a local optimum (local optimizer), poor performance when the number of dimensions grows, and sensitivity of the algorithm to the rotation of the search space. The significance of each of these issues was discussed and it was argued that none of the particle swarm optimizers they are aware of can address all of these issues at the same time. To address all of these issues at the same time, a new general form of velocity update equation for the PSO algorithm, named as BPSO, that contains a user-definable normal distribution function around local and global memories was proposed. It is proven that the proposed velocity update equation guarantees to address all of mentioned issues satisfactorily.

In a reasonable fashion, deterministic gradient-based approaches are further detailed in Sections 2.6 and 2.7 and new approaches, named as SAPSO and ISAPSO algorithms, are later reported in Chapter 3 where important features, such as gradient information, diversity control and rotation matrices, are embedded in all particles' motion at each iteration of the search process.

1.4 Motivation

The literature of swarm intelligence, bio- and nature-inspired metaheuristics is wide and still growing (BOUSSAïD; LEPAGNOT; SIARRY, 2013; BLUM et al., 2011; SHEHAB; KHADER; AL-BETAR, 2017; HOSSEINI; KHALED, 2014). The trend seems to be the use of any kind of real-world observation as an approach to face optimization problems. The population-based metaheuristics related to some animal wildlife-behavior observation have a broad and miscellaneous list. To mention some bio-inspired algorithms for optimization: Ants (DORIGO; MANIEZZO; COLORNI, 1996), Bees (KARABOGA, 2005), Glowworms (KRISHNANAND; GHOSE, 2005) or Fireflies (YANG, 2008; YANG, 2009), Frogs (EUSUFF; LANSEY; PASHA, 2006), Cuckoos (BLUM et al., 2011), Bats (YANG; HE, 2013), Fishes (QIAN, 2002), Spiders (YU; LI, 2015), Whales (BIYANTO et al., 2017), and so on. The algorithms related to nature-inspired observation are even more creatives. To name a few: River Formation Dynamics (RABANAL; RODRÍGUEZ; RUBIO, 2007), Law of Gravity and Mass Interactions (RASHEDI; NEZAMABADI-POUR; SARYAZDI, 2009), Flower Pollination (YANG, 2012), Hydrological Cycle (WEDYAN; WHALLEY; NARAYANAN, 2017), Mine Bomb Explosion (SADOLLAH et al., 2013), Bib-Bang Crunch (ZANDI; AFJEI; SEDIGHIZADEH, 2012), Electromagnetism (CUEVAS et al., 2012), and so on. The work made by Fister Jr (JR. et al., 2013) has a review of the well-known metaheuristics.

One might say is challenging to find papers with breakthrough results in the optimization area, due to the constant "reinvention of the wheel", i.e., the authors use different terms for several common concepts and call them new (SÖRENSEN, 2015). This fact negatively contributes to the area of optimization and deviates the attention to other spotlights, instead of solving at least one of the optimization challenges, for instance, to highlight a few of them related to population-based approaches: slow convergence process, re-visitation of areas already evaluated, prominent areas of the search space not properly investigated, the algorithms are prone to premature convergence specially when they are highly dependent of their initial parameters, the trade-off between rotation invariance and directional diversity, and biases toward the coordinate axes.

This thesis is pursuing to face some of the aforementioned challenges in the literature of global continuous optimization: to properly investigate areas of the search space with

gradient information, to avoid premature convergence with the diversity control mechanism, to develop PSO variants which satisfy the property *rotation invariance* and have directional diversity simultaneously. Moreover, the algorithms developed in this thesis are under a rigorous methodology of evaluating metaheuristic approaches (CLERC, 2012a), including the analyses of multiple performance criteria related to stochastic variables. Besides, the analyses described in this thesis are also based on a reliable number of algorithm's run whose results are later submitted to statistical hypothesis tests, such as Friedman's and Wilcoxon's tests.

1.5 Justification

This thesis will later describe in detail the foundations of SAPSO algorithm with deterministic characteristics, an empirical analysis about rotation and information exchange among particles, and a final algorithm called ISAPSO built by a consistent aggregation of all previous valuable technical contributions, which embodies prominent features from SAPSO and the property of *rotation invariance*. The enhancements provided by this thesis are worth since the essence of such innovations in the original works has some fundamental issues discussed here.

As previously presented, the authors of the works (NOEL; JANNETT, 2004; NOEL, 2012) seem to be the first ones to build a gradient-based PSO algorithm. The former work embedded the gradient information in the velocity update equation, while the later boosted the initial proposal by applying the gradient descent algorithm in the current global best position of the swarm only. Invariably, those approaches introduced a deterministic direction jointly with the global tendency of the swarm. Although better results were reported by the numerical simulations, the methodology seems not convincing, as the authors evaluated the algorithms in a set of simple benchmark functions without concerning about statistical significance analysis. Furthermore, in the second work, the idea sounds like a memetic approach, once the intention is to improve the global best position with a few iterations of the deterministic method. This poses a fundamental issue of erroneously push the swam to local minima without any possibility to escape. This is also the same problem found in the following works (ZAHARA; KAO; SU, 2009; ZAHARA; KAO; LIU, 2009; LIU; HAN, 2013; HAN; LIU, 2014; HAN; LIU, 2015). In (HAN; LIU, 2015), for instance, the work proposed an adaptation of the previous work (LIU; HAN, 2013), which uses a second order derivatives with Quasi-Newton method to move particles in the search space. The particle's position is only updated when a new position is better than the old one, which increases the computational cost of the algorithm and has the potential to waste a such important deterministic direction provided by Newton's method.

The aforementioned modifications safely provide interesting approaches to the

literature of PSO. However, using gradient-based information significantly increases computational costs. One may think that gradient direction must be used wisely, instead of using as a possibility for moving particles. Consequently, SAPSO and ISAPSO algorithms work with semi-autonomous particles, i.e., each particle decides between two possible types of movements: follow the global tendency or its local derivatives. This decision equips each particle with the possibility to solely escape from local minima. In addition, the diversity control is another mechanism to avoid premature convergence by repelling particles whenever necessary.

The random walk mainly presented in both random weighted terms of velocity update equation can compromise the search process and lead the swarm to an explosion, also termed as *drunkard's random* (OZCAN et al., 1999; CLERC; KENNEDY, 2002). Some mechanisms to decrease this effect by controlling the dynamics of the inertia weight, cognitive and social coefficients are present in the literature (SHI; EBERHART, 1999; RATNAWEERA; HALGAMUGE; WATSON, 2004). In this thesis, different mechanism to control the dynamics of particles are investigated. SAPSO algorithm use static parameters to control cognitive and social coefficients, while the inertia weight value decreases linearly throughout iteration. In contrast, ISAPSO algorithm simplifies parameter settings by fixing the three parameters with the same well-known values of the literature mainly based on (CLERC, 2012b). This thesis also reserves a deep analysis about four combinations of PSOs. A rotationally variant and invariant coupled with a fast and late exchange information among particles are evaluated on a benchmark functions to investigate whether the presence or absence of those properties signifies different performance when considering different and complex test optimization problems. The prominent results of this study motivated the use of rotation matrices in ISAPSO algorithm, previously justified by the works (WILKE; KOK; GROENWOLD, 2007b; BONYADI; MICHALEWICZ, 2014).

To summarize important characteristics present in each related work described so far, Table 1 categorizes the algorithms by the following features: cognitive coefficient (c_1), social coefficient (c_2), inertial weight (w), and constriction factor (χ) can be static (same values for all types of problem) or dynamic (varying along iterations or problem-dependent); diversity control (dir), gradient information (∇), rotation matrix (rm), rotation invariance (inv), and directional diversity (dd). The last two rows show the SAPSO and ISAPSO approaches described later in this thesis. At first glance, SAPSO algorithm is similar to the work (HAN; LIU, 2014), but with dynamically c_1 (social coefficient in this context) and c_2 (gradient coefficient). Despite this, DGHPSOGS actually uses two algorithms in its formulation, one is used to crowd particles and the other one to disperse, which brings complexity to the implementation of the approach. Differently, the gradient information and diversity control is embedded in just one algorithm with SAPSO. Moreover, there are fundamental issues related to the velocity update equations and the diversity control in DGHPSOGS algorithm, which consumes computational time and a fast switching

Table 1 – Principal features of each related work.

Algorithm	c_1	c_2	w	χ	*div*	∇	*rm*	*inv*	*dd*
(KENNEDY; EBERHART, 1995)	static	static	-	-	✘	✘	✘	✔	✘
(EBERHART; KENNEDY, 1995)	static	static	-	-	✘	✘	✘	✔	✘
(KENNEDY, 1997b)	static	static	-	-	✘	✘	✘	✔	✘
(SHI; EBERHART, 1998a)	static	static	dynamic	-	✘	✘	✘	✔	✘
(SHI; EBERHART, 1999)	static	static	dynamic	-	✘	✘	✘	✔	✘
(CLERC, 1999)	static	static	-	static	✘	✘	✘	✔	✘
(EBERHART; SHI, 2000)	static	static	dynamic	static	✘	✘	✘	✔	✘
(CLERC; KENNEDY, 2002)	static	static	-	static	✘	✘	✘	✔	✘
(VESTERSTRØM; RIGET, 2002) – ARPSO	static	static	dynamic	-	✔	✘	✘	✔	✘
(YAMAGUCHI; YASUDA, 2006)	dynamic	dynamic	static	-	✘	✘	✘	✘	✔
(ZHAN et al., 2007)	dynamic	dynamic	static	-	✘	✘	✘	✘	✔
(ZHAN et al., 2009)	dynamic	dynamic	dynamic	-	✘	✘	✘	✘	✔
(NOEL, 2012) – GPSO	static	static	-	-	✘	✔	✘	✔	✘
(HAN; LIU, 2014) – DGHPSOGS	static	static	dynamic	-	✔	✔	✘	✔	✘
(WILKE; KOK; GROENWOLD, 2007b) – WPSO	static	static	static	-	✘	✘	✔	✔	✔
(CLERC, 2012b) – SPSO	static	static	static	-	✘	✘	✘	✔	✔
(BONYADI; MICHALEWICZ; LI, 2014)	static	static	static	-	✘	✘	✔	✔	✔
(BONYADI; MICHALEWICZ, 2014) – BPSO	static	static	static	-	✘	✘	✘	✔	✔
(SANTOS et al., 2018) – SAPSO	dynamic	dynamic	dynamic	-	✔	✔	✘	✔	✘
ISAPSO algorithm	static	static	static	-	✔	✔	✔	✔	✔

Source: produced by the author.

between both algorithms. ISAPSO algorithm turns every coefficient to static variables independently of the optimization problem under consideration. Thus, the nuisance about problem-dependent parameter is overcome by this approach. Furthermore, the rotation matrices are introduced in the velocity update equation to guarantee directional diversity of search distribution and to satisfy the property *rotation invariance* (i.e., to obtain similar performances regardless of the coordinate system) at the same time.

1.6 Objectives

The main objective of this thesis is to improve the global search process of PSO algorithm by introducing deterministic gradient-based information along with the properties *rotation invariance* and *directional diversity*. A mathematical proof is given in order to show it is possible to couple both properties at the same time and to yield a general PSO version that can be applied in a larger class of optimization problems. A rotationally invariant PSO version, coined as ISAPSO, is the final outcome of this thesis.

The following specific objectives are addressed to achieve the main objective:

- To embed the gradient direction into the classical PSO. The expectation is to find local optima of the objective function without wasting computational time, since gradient is a deterministic information that points in the direction of the greatest rate of increase (or decrease) of the function;

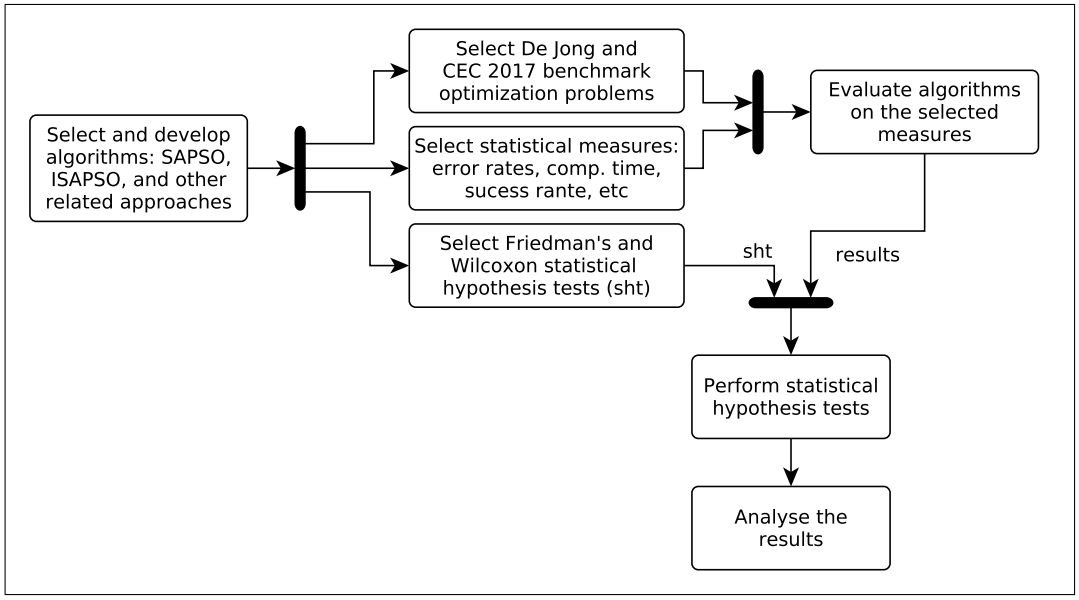
- To incorporate the diversity controlling approach into the classical PSO. This objective faces the premature convergence normally present in any metaheuristic;
- To test the enhanced PSO version in a set of benchmark functions. The purpose is to compare the performance of the proposed approach with other related PSOs;
- To evaluate the enhanced PSO version through rigorous statistical hypothesis tests;
- To develop a simple method to define the minimum number of algorithm runs to consider reliable results. This method must be able to work with one or more optimization problems. Moreover, performance criteria such as success rate or objective function value must be suitable to work with;
- To provide empirical analyses of rotation and information exchange among particles. The assumption to be addressed is whether both subjects impact on the performances of classical PSO versions.

1.7 Methodology

A common methodology to evaluate multiples metaheuristic approaches is depicted in Figure 1. After selecting and developing the discussed algorithms, such as SAPSO and ISAPSO, two benchmark optimization problems are separated to later evaluate the non-deterministic approaches. In this thesis, the enhancements are being subjected to a suite of test functions based on De Jong's problems (JONG, 1975) and CEC 2017 benchmark optimization problems (AWAD et al., 2017), which is formed by unimodal, multimodal, hybrid and composition test functions. Each test function offers different challenges for any optimization algorithm, such as: many local minima surrounding the global minimum, nearly flat regions, high dimensional continuous space for searching, decentralized global minimum, among others. The functions are used to assert the qualities of the algorithms, particularly on multimodal functions where the algorithms must be able to avoid local minima and fine-tune the global optimum. Besides, modern modifications provide even more challenging test function for metaheuristic approaches, since shifted, rotated and composed functions are available to test the different features of the algorithms.

To evaluate multiples metaheuristics, it is common to consider many executions of the algorithm applied on a specific test functions and average the results based on different statistical measures, which reduces the probability to reach good results by chance. Thus, estimating a reliable number of algorithm's execution is very important and if the number is not properly defined, the comparisons might be impaired. A method to estimate a reliable number of executions is formulated in this thesis based on the law of large numbers and Chebyshev's Inequality, and the technique is used throughout the experimental results. All approaches deployed in this thesis are evaluated in this manner.

Figure 1 – General methodology of the thesis.



Source: produced by the author.

The principal measurements used for comparison among algorithms are highlighted in the following:

- Minimum value found: this is related to the function value found by the best particle of the swarm when the algorithm stops.
- Error rate: it is related to the difference between the minimum function value found by the swarm and the true global optimum value of the objective function.
- Success rate: it is a performance criterion to determine the achievement of an algorithm, i.e., whether the goal of the optimization problem is found or not. This measurement is binary, resulting in 1 if the desired error rate is successful achieved, or 0 otherwise. The desired error rate has to be below a predefined threshold value (e.g., 10^{-2}). Although being a binary variable, multiple executions of the algorithms transform the success rate into percentage values between $[0, 100]$, which is a value to be used by a statistical hypothesis test due to the bounded values.
- Computational time: a time elapsed required to run all algorithmic steps. This measured performance gives a different perspective of the obtained results, as the main goal is to evaluate the faster approach.
- Required iterations: a measure to handle the number of iterations required to stop the algorithm. This is usually related to the number of iterations required to achieve convergence, i.e., the exact iteration where the algorithm finds the global minimum

of the test function or the approximated global best result ruled by a predefined threshold.

- Statistical hypothesis test: a non-parametric test, such as Friedman's and Wilcoxon's tests, aims to detect significant differences between two sample means and can be applied to continuous data through a ranking-based transformations (DERRAC et al., 2011; HOLLANDER; WOLFE; CHICKEN, 2013). These tests also produce the p-value, i.e., the probability, given the null hypothesis, of obtaining a result equal to or more extreme than what was actually observed.

Besides classical statistical measures, such as error and success rates, Friedman's and Wilcoxon's hypothesis tests are the ones selected to evaluate the statistical significance of the outcomes. These statistical inferences report when an algorithm is way better than other statistically, considering a set of optimization problems. Finally, the following steps of the general methodology are sequential application of the selected components. In the end, an analysis of the results is left for the practitioner.

1.8 Thesis organization

The remainder of this thesis is organized as follows. In Chapter 2, a literature review of the classical PSO algorithm along with the most significant improvements made by other authors is presented. The gradient descent, Newton's and Quasi-Newton's deterministic algorithm for optimization are also explained. Furthermore, the mathematical tool to prove that an algorithm is scale, translation and rotation invariant is provided. Enhancements provided by this thesis are explained in details in Chapter 3. This chapter also brings an important discussion about exploration versus exploitation in the field of metaheuristics; influence of rotation and information exchange among particles; a mathematical proof that ISAPSO is rotationally invariant is given. In Chapter 4, the simulation results of all proposed enhancements are presented. Finally, Chapter 5 shows the concluding remarks of this thesis. This chapter also discusses about future researches.

2 Theoretical background

This chapter presents the theoretical background that will be widely used throughout this thesis. The author provides a notation to explain the classical PSO algorithm along with the two main designs to spread information among particles. The heuristics about bound handling and velocity clamp are discussed next. This chapter also reports the significant improvements in the classical PSO algorithm since its first version in 1995. The roots of gradient descent algorithm, Newton's and Quasi-Newton's methods are formulated hereafter. Later, the mathematical tool to prove that an algorithm is scale, translation and rotational invariant is exposed in detail. After, the versions of PSO related to this thesis are described. This chapter ends with Maurice Clerc's rules and a general formulation of optimization problems.

2.1 The classical particle swarm optimization

The classical particle swarm optimization, a global search algorithm firstly introduced by Kennedy and Eberhart (KENNEDY; EBERHART, 1995; EBERHART; KENNEDY, 1995), has roots in the social behavior of bird flocking and fish schooling originally published by Reynolds (REYNOLDS, 1987a; REYNOLDS, 1987b), Heppner (HEPPNER; GRENANDER, 1990) and Wilson (WILSON, 1975). The idea of particle swarm arose through simulation of a simplified social model which is intrinsically based on a social metaphor, though the algorithm stands without metaphorical support.

Reynolds was pursuing to understand the aesthetics of bird flocking, whereas Heppner was interested in discovering the underlying rules that enable large numbers of birds to flock synchronously, often changing direction suddenly, scattering and regrouping. Both scientists came out with the idea that a cellular automata might underlie the unpredictable group dynamics of bird social behavior, relying heavily on manipulation of inter-individual distances and the synchrony of flocking behavior might be a function of birds' efforts to maintain an optimum distance between themselves and their neighbors.

The crucial mechanism of cooperation among particles presented in the PSO algorithm comes from another source, the sociobiologist Wilson (WILSON, 2000 apud KENNEDY; EBERHART, 1995) once said about fish schooling: "*In theory at least, individual members of the school can profit from the discoveries and previous experience of all other members of the school during the search for food*". In other words, the social sharing of information among herds, schools, flocks, and humans offers an evolutionary advantage when cooperating with each other, instead of competing for food items.

The PSO algorithm requires only primitive mathematical operations and is computationally inexpensive in terms of memory and speed. One of the main benefits of PSO is its simplicity and the power for solving complex optimization problems, avoiding the use of any sophisticated evolutionary operations found in GAs (GOLDBERG, 1989). Also, due to the metaheuristic nature and easy implementation, PSO has been used in many domains of science such as training neural network weights and select the best network architecture (GUDISE; VENAYAGAMOORTHY, 2003; GRIMALDI et al., 2004), text feature selection (LU et al., 2015), time series forecasting (CHOUIKHI et al., 2017), dynamic vehicle routing (OKULEWICZ; MAńDZIUK, 2017), and many others.

The main idea of PSO is to perform a biased stochastic search of the global optimum solution through the search space of a problem. However, the random walk might lead to premature convergence like most of the stochastic methods, specially in multimodal optimization problems where the fitness landscape is quite irregular and challenging to find the optimal solution. This is one of the challenges that the new generation of metaheuristic approaches must be concerned about. One way to deal with this challenge is to diminish the probability of unfeasible results or to provide mechanisms to vanish the problem of premature converge, either would be of great advance in the field of metaheuristics.

The PSO algorithm is a population- or swarm-based technique compound by particles. Each particle is a candidate solution to the optimizing problem in a d-dimensional space. The position of the ith particle is denoted as $\vec{\mathbf{x}}_i = [x_{i1}, x_{i2}, \ldots, x_{id}]$, $i = 1, 2, \ldots, n$, where n represents the number of particles. The ith particle also "remembers" its previous best position denoted as $\vec{\mathbf{p}}_i = [p_{i1}, p_{i2}, \ldots, p_{id}]$. The previous best position, also known as "local memory", is the one with the best objective function value found so far by the ith particle. The global best position of the swarm is expressed as $\vec{\mathbf{g}} = [g_1, g_2, \ldots, g_d]$. This position is termed as "global memory" and is considered herein as the best local memory among the particles. The previous and global best positions are computed in relation to the iteration t, respectively, as

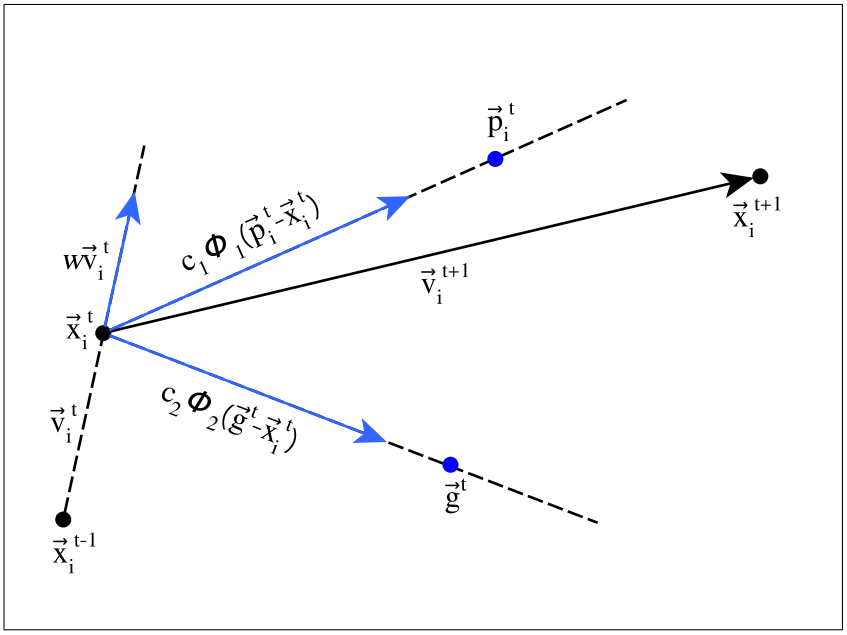
$$\vec{\mathbf{p}}_i^t = \arg\min_{\vec{\mathbf{x}}_i^k} \{f(\vec{\mathbf{x}}_i^k) \mid k = 1, 2, \ldots, t\} \tag{2.1}$$

and

$$\vec{\mathbf{g}}^t = \arg\min_{\vec{\mathbf{p}}_i^t} \{f(\vec{\mathbf{p}}_i^t) \mid i = 1, 2, \ldots, n\}. \tag{2.2}$$

Note that, herein the swarm considers a fully connected topology, i.e., one local memory must be strictly a global memory at each iteration. Each particle of the swarm has a velocity vector denoted as $\vec{\mathbf{v}}_i = [v_{i1}, v_{i2}, \ldots, v_{id}]$. The classical PSO algorithm (KENNEDY; EBERHART, 1995) updates each particle's position by a random weighted average between the previous best position found so far and the global best position of

Figure 2 – PSO model of a particle's motion.



Source: produced by the author.

the entire population, defined as

$$\vec{\mathbf{v}}_i^{t+1} = \vec{\mathbf{v}}_i^t + \underbrace{c_1\phi_1\left(\vec{\mathbf{p}}_i^t - \vec{\mathbf{x}}_i^t\right)}_{\text{cognitive component}} + \underbrace{c_2\phi_2\left(\vec{\mathbf{g}}^t - \vec{\mathbf{x}}_i^t\right)}_{\text{social component}} \tag{2.3}$$

and

$$\vec{\mathbf{x}}_i^{t+1} = \vec{\mathbf{x}}_i^t + \vec{\mathbf{v}}_i^{t+1}, \tag{2.4}$$

where c_1 and c_2 correspond respectively to the cognitive and social coefficients; ϕ_1 and ϕ_2 are two uniformly distributed random numbers between $[0, 1]$. A visual representation of a particle's movement at instant t is depicted in Figure 2.

One can see three forces acting over the particle $\vec{\mathbf{x}}_i^t$, such that:

1. $\vec{\mathbf{v}}_i^t$: the momentum associated with the last velocity vector of the ith particle, from $\vec{\mathbf{v}}_i^t = \vec{\mathbf{x}}_i^t - \vec{\mathbf{x}}_i^{t-1}$. The last velocity $\vec{\mathbf{v}}_i^t$ is used as an inertial direction at iteration t, i.e., the direction where the ith particle was going in the last iteration;

2. $c_1\phi_1\left(\vec{\mathbf{p}}_i^t - \vec{\mathbf{x}}_i^t\right)$: this force is about the importance given to the local memory of the particle; the quantity of knowledge the ith particle is taken from its local memory. The cognitive coefficient c_1 plays an important role in this matter and is set by the practitioner. Note that values of $c_1 > 1$ means the ith particle might fly over the $\vec{\mathbf{p}}_i$ position sometimes during the iterations;

3. $c_2\phi_2\left(\vec{\mathbf{g}}^t - \vec{\mathbf{x}}_i^t\right)$: this component is associated with the global memory of the swarm, i.e., the direction to reach the global best position regarding the current $\vec{\mathbf{x}}_i^t$ position.

One can note this component shares the global memory with the whole swarm. This is the component associated with the crowd of particles in some region of the search space in the last iterations. The parameter c_2 controls the speed at which global memory information is transmitted among the particles; When $c_2 > 1$, the ith particle might overpass the $\vec{\mathbf{g}}^t$ position during algorithm's run.

The search process of a classical PSO is presented in Algorithm 1. Lines 2–3 initialize all structures and obtain the best position among all particles, respectively. A common form to initialize the particles is to uniformly distribute them through out the search space with a uniformly distributed random number generator applied at each dimension of each particle. The velocity of each particle is initialize at zero and the local memory of each particle starts with its own position at the beginning. The global memory must be obtained according to the Equation 2.2 with $\vec{\mathbf{p}}$ or $\vec{\mathbf{x}}$ as argument, since $\vec{\mathbf{p}} \leftarrow \vec{\mathbf{x}}$ at the beginning.

From lines 4–15, the algorithm runs the main loop. When a stop criterion matches the predefined practitioner's criteria, then the main loop stops. Common examples of stop criteria can be: exceeded the maximum number of iterations; found the global best solution (in real-world engineering application this stop criterion may not be applied, as the best solution is often not known); a sub-optimal solution $\vec{\mathbf{x}}$ was found, such that a predefined error ϵ was achieved according to $|f(\vec{\mathbf{x}}) - f(\vec{\mathbf{x}}_{opt})| < \epsilon$; detected a premature convergence state of the swarm; a number of successive iterations without improvements in the overall fitness of the swarm; exceeded the maximum number of objective function calls.

The inside loop (lines 5–14) updates the velocity, position and fitness of each particle, and also checks whether the new position is better than the memories (lines 8

Algorithm 1 Classical PSO algorithm.

```
1:  function PSO(x⃗, v⃗, p⃗)
2:      initialize x⃗, v⃗, p⃗ ← x⃗        ▷ initialize positions at random and velocities at 0
3:      g⃗ ← GETBEST(p⃗)                  ▷ get global best position (Equation 2.16)
4:      while stop criteria not reached do
5:          for i = 1 to n do
6:              v⃗_i ← UPDATE(x⃗_i, v⃗_i, p⃗_i, g⃗)   ▷ update particle velocity (Equation 2.3)
7:              x⃗_i ← x⃗_i + v⃗_i                 ▷ update particle position (Equation 2.4)
8:              if f(x⃗_i) < f(p⃗_i) then
9:                  p⃗_i ← x⃗_i                   ▷ update the previous best position
10:                 if f(x⃗_i) < f(g⃗) then
11:                     g⃗ ← x⃗_i                 ▷ update the global best position
12:                 end if
13:             end if
14:         end for
15:     end while
16:     return g⃗
17: end function
```

and 10). A *true* statement in the line 8 indicates the new position becomes the previous best position of the particle found (line 9). Finally, it checks whether the global best position needs to be updated (line 10–12). This if-then statement applies the Equation 2.2 online, i.e., right after each particles' movement, the global memory can be updated if necessary. The result of the algorithm is the global best position (line 16). This described PSO version is known as rotationally invariant with fast exchange information among particles. More information about this nomenclature is given in Sections 2.2 and 2.8.

2.2 Information exchange among particles

The vectors $\vec{\mathbf{p}}$ and $\vec{\mathbf{g}}$ are commonly referred to as local and global memory, respectively. Local memory forces the particle towards its previous best position, which indicates a local exploitation performed by each particle. Global memory is another force that pushes the particle, and the swarm as well, in the direction of the global best position found so far by the population. The latter is the only information exchange among particles while the algorithm is running.

Some versions of PSO attempts to update the global memory after the movement of an ith particle, such that the $(i+1)$th particle uses this new global memory immediately (see Algorithm 2). Other versions await an entire iteration to attempt to update global memory (see Algorithm 3). The central issue is that a small difference can interfere the results, by decreasing or increasing the computational time require to run the algorithm and affecting the convergence of the swarm, or even precluding the parallelism of the algorithm.

Algorithm 2 Fast information exchange.

```
1: while stop criteria not matched do
2:     for i = 1 to n do
3:         ...
4:         if f(x_i) < f(g) then
5:             g ← x_i                          ▷ update the global best position
6:         end if
7:         ...
8:     end for
9: end while
```

Algorithm 3 Late information exchange.

```
1: while stop criteria not matched do
2:     for i = 1 to n do
3:         ...
4:     end for
5:     g ← GETBEST(p)                ▷ get global best position (Equation 2.2)
6: end while
```

2.3 Bound handling

In most computing and engineering applications, the optimization problem must be bound, i.e., $\vec{\mathbf{x}} \in \mathbb{S} \mid \mathbb{S} \subset \mathbb{R}^d$, for preventing the roam of the particles to infinity throughout the search space. One of the most important variant of constraints is called *box constraints*, which is expressed in the form

$$\forall j \in \{1, 2, \ldots, d\}, \forall i \in \{1, 2, \ldots, n\} : \; x_{min} \leq x_{ij} \leq x_{max},$$

the range of each variable x_{ij} has a lower bound x_{min} and an upper bound x_{max}. When a particle moves outside of the box constraints, some strategy must be adopted to push back the particle inside the feasible area. For constraint optimization problems of this form, a large number of constraint handling mechanisms are available in the literature. Figure 3 depicts four different box constraint-handling methods described below:

- Infinity (Figure 3a) (HU; EBERHART, 2002): the objective function values for particles outside the boundaries are considered infinity;
- Absorption (Figure 3b) (CLERC, 2006): a scale factor is used to shrink the length of the updated velocity, so as the particle's position ends up on the boundary;
- Nearest (Figure 3c) (CLERC, 2006): if a particle leaves the space of the feasible solutions, then it is set to the nearest position inside the box constraint;
- Reflection (Figure 3d) (BOCHENEK; FORYŚ, 2006): a reflection at the border is applied in cases where the particle ends up outside the boundaries.

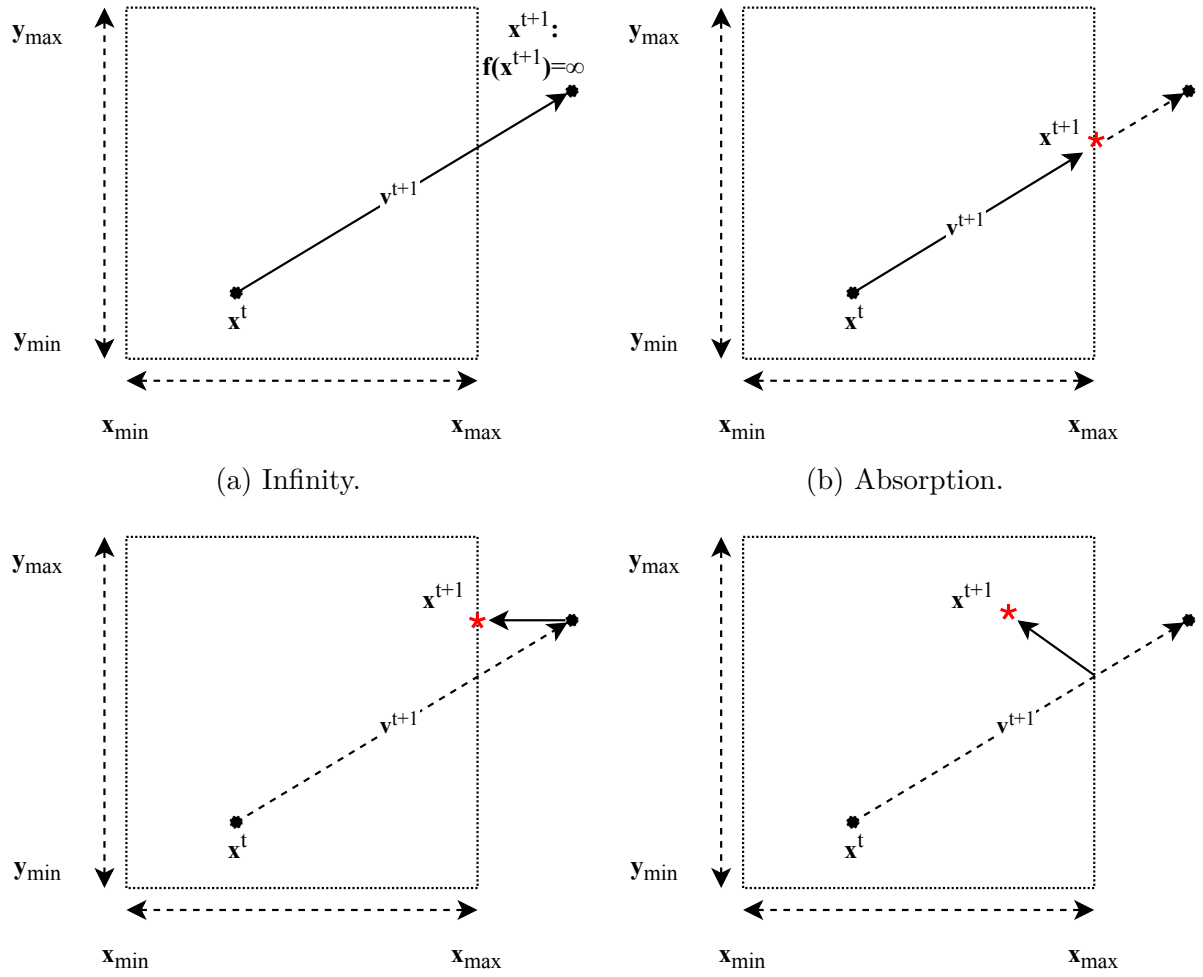
2.4 Velocity clamping

A common velocity clamping is to limit the velocity of the particles to a specific value v_{max} to avoid that the magnitudes of the particles' positions and velocities increase in an uncontrolled manner (BRATTON; KENNEDY, 2007). To achieve this, it is necessary to apply right after the standard velocity update Equation 2.3 the following condition

$$v_{ij} = \begin{cases} \text{sign}(v_{ij}) v_{max} & \text{if } |v_{ij}| \geq v_{max} \\ v_i & \text{else} \end{cases}, \tag{2.5}$$

where sign$(\cdot)$ is the signal function. Later in this chapter, a discussion about the constriction coefficients is elaborated based on a deterministic PSO model described by Clerc and Kennedy (CLERC; KENNEDY, 2002). They showed that a restriction of the particles' velocities is not necessary to obtain a convergent particle swarm, but as a price, the velocity update equation must use a constriction factor.

Figure 3 – Box constraint-handling methods.



(a) Infinity. (b) Absorption.

(c) Nearest. (d) Reflection.

Source: produced by the author.

2.5 Improvements on particle swarm optimization

Since the first release of the PSO algorithm developed by Kennedy and Eberhart in 1995, some significant improvements were provided by the scientific community. In this section, some well-known improvements embedded in the PSO algorithm are discussed. One important consideration to make is that the "significant improvements" herein are related to the breakthrough ideas, i.e., innovative concepts used by many practitioners and in different computing and engineering optimization problems.

2.5.1 Inertia weight

The paper published by Shi and Eberhart (SHI; EBERHART, 1998a) introduced a new parameter, called the *inertia weight*, into the original particle swarm optimizer. This parameter plays an important role balancing the trade-off between global and local

search. It can be a positive constant or even a positive linear or nonlinear function of time. Equation 2.3 was updated by the following

$$\vec{\mathbf{v}}_i^{t+1} = w\vec{\mathbf{v}}_i^t + c_1\phi_1\Big(\vec{\mathbf{p}}_i^t - \vec{\mathbf{x}}_i^t\Big) + c_2\phi_2\Big(\vec{\mathbf{g}}^t - \vec{\mathbf{x}}_i^t\Big), \tag{2.6}$$

where w stands for the inertia weight. According to the experimental tests, when the inertia weight is small (e.g., $w < 0.8$) the PSO algorithm acts like a local search algorithm, whereas when the inertia weight is large (e.g., $w > 1.2$), the PSO algorithm resembles a global search method and tries to explore new areas. A moderate value between $[0.9, 1.2]$ is a good range to choose w from. However, one can note the discussions presented in the original paper were based on a single multimodal function. Therefore, any choice of w must be considered with care.

A time decreasing inertia weight is also introduced in the same paper which ended up in a significant improvement on the PSO performance. This paper was the first one to admit a function over time to update the inertia weight parameter. Although the idea might be simple at first sight, it has opened a large area of study regarding the update of time-varying parameters.

2.5.2 Dynamic adjustments for inertia weight

The notion about linear decreasing inertia weight arose in the paper (SHI; EBERHART, 1998a). Ever since the papers (SHI; EBERHART, 1998b; SHI; EBERHART, 1999; SUGANTHAN, 1999) matured the proposal of partially control the search ability of the swarm and the PSO algorithm itself. Note that this approach was later termed as time-varying inertia weight (RATNAWEERA; HALGAMUGE; WATSON, 2004; ARUMUGAM; RAO, 2006).

The inertia weight w is employed to control the impact of the previous history of velocities on the current velocity of the particle, thereby influencing the trade-off between global (wide-ranging) and local (nearby) exploration abilities of the particles. A larger inertia weight facilitates global exploration (searching new areas), while a smaller inertia weight tends to facilitate local exploration to fine-tune the current search area. One can think that the inertia weight is a problem-dependent parameter. However, a suitable selection of the inertia weight is more related to the balance between global and local exploration abilities, than being a problem-dependent parameter. Equation 2.7 provides a simple form to achieve the linear decreasing inertia weight,

$$w^t = w_1 - \Big(w_1 - w_2\Big) \times \frac{t}{T}, \tag{2.7}$$

where w_1 and w_2 are the initial and final inertia weights, and T is the maximum number of iterations.

The paper (SUGANTHAN, 1999) illustrates a number of techniques to improve the standard PSO algorithm. One of the improvements is the same previously presented in the work (SHI; EBERHART, 1998b), which is the time-varying the inertia weight. The author states that the magnitudes of the random walk and inertia weight in the PSO can be gradually adjusted to perform a fine grain search during the final stages of optimization. The proposal is evaluated in set of benchmark functions. Another work (SHI; EBERHART, 1999) extensively tested the PSO algorithm with linearly decreasing inertia weight by experimental studies of four non-linear functions. The experimental results illustrated that the PSO may lack global search ability at the end of a run due to the utilization of a linearly decreasing inertia weight. However to some extent, this can be overcome by employing a self-adapting strategy for adjusting the inertia weight. A comparative study carried out in (BANSAL et al., 2011) on a set of benchmark functions for optimization has shown that random inertia weight is suitable if faster convergence is desired for the practitioner. Some of the strategies are presented as follows

$$w^t = 0.5 + \frac{r}{2}, \tag{2.8}$$

$$w_{ij}^t = 1.1 - \frac{\vec{\mathbf{p}}_{ij}}{\vec{\mathbf{g}}_j}, \tag{2.9}$$

$$w^t = (w_1 - w_2)\left(\frac{T-t}{T}\right) + w_2 z, \tag{2.10}$$

$$w^t = 0.5r + 0.5z, \tag{2.11}$$

$$w^t = w_1 - (w_1 - w_2) \times \left(\frac{t}{T}\right)^3, \tag{2.12}$$

where r is a uniformly distributed random number between $[0, 1]$, and $z = 4r(1-r)$. The Equations 2.8 (EBERHART; SHI, 2001), 2.9 (ARUMUGAM; RAO, 2006), 2.10 (FENG et al., 2007), 2.11 (FENG et al., 2007), and 2.12 (NAKAGAWA; ISHIGAME; YASUDA, 2008) are somewhat reinforcing the importance of the inertia weight and trying to determine the contribution rate of a particle's previous velocity to its velocity at the current time step.

2.5.3 Constriction factor

A *constriction coefficient* χ was also introduced in a study about explosion, stability and convergence of particle swarm (CLERC; KENNEDY, 2002), which ensures convergence of the entire population overtime. The principal modifications are expressed as

$$\chi = \begin{cases} \frac{2\kappa}{\psi - 2 + \sqrt{\psi^2 - 4\psi}} & \text{if } \psi > 4;\ \psi = c_1 + c_2 \\ \kappa & \text{else} \end{cases}; \quad 0 < \kappa < 1 \tag{2.13}$$

and

$$\vec{\mathbf{v}}_i^{t+1} = \chi\Big(\vec{\mathbf{v}}_i^t + c_1\phi_1\big(\vec{\mathbf{p}}_i^t - \vec{\mathbf{x}}_i^t\big) + c_2\phi_2\big(\vec{\mathbf{g}}^t - \vec{\mathbf{x}}_i^t\big)\Big). \tag{2.14}$$

The application of constriction coefficients allows control over the dynamical characteristics of the particle swarm, including its exploration versus exploitation. The κ value allows the practitioner to control the degree of convergence by setting to various values between $[0, 1]$, while a common $\psi = 4.1$, implies $\chi = 0.729$. Another important statement of the aforementioned paper is that the random weighting of the control parameters in the PSO algorithm results in a kind of explosion or a "drunkard's walk" as particles' velocities and positions careen toward infinity. Traditionally, the explosion has been faced with v_{max} parameter, which limits the velocity.

The authors also demonstrated that the particle swarm algorithm can be conceived of in such a way that the explosion can be controlled, without resorting to the definition of any arbitrary or problem-specific parameters. The implementation of properly defined constriction coefficients can prevent explosion, as well as the model can be parameterized in such a way that the particle system consistently converges on local optima. The last reminder of the authors is that the real strength of the particle swarm derives from the collaboratively interactions among particles as they search the space. The third term from the velocity update equation, termed as "social influence" is derived from the successes of other particles, such that if this influence is removed, the PSO algorithm's performance is abysmal (KENNEDY, 1997a). As a particle swarm searches during run time, particles are moving toward one another's successes. An usual result is a cluster of particles in optimal regions of the search space.

2.5.4 PSO models of velocity equation

Kennedy's work (KENNEDY, 1997a) highlights some further details about the influence of each term of the velocity update equation. The author considers a neural network XOR problem to be solved by the particle swarm. Experimental tests evaluated four different models of PSO: 1) Full model: uses the classical velocity update equation to optimize a problem (Equation 2.3); 2) Cognitive-only model: removes the third term of the equation (social influence); 3) Social-only model: takes away the second term (cognitive influence); and 4) Selfless model stands for a variation of the social-only model where the ring topology does not consider the ith particle itself.

The study showed that in the social-only model, the algorithm tended to be a more efficient optimizer for the present XOR problem than both cognitive-only and the full models of PSO, whereas in the cognitive-only model the particles are prone to search areas in which they had been initialized and failed to move into optimal regions. As the selfless model is just a variation of the social-only model, one can see little improvements in the

PSO algorithm regarding the full and cognitive-only model, but inferior when compared with the social-only model.

One important mention is that the results represent performances of different versions of the algorithm on one particular problem. Obviously, the models of velocity will certainly perform differently on problems featuring higher dimensionality, non-linearity, multimodal, and so on.

The proposed SAPSO and ISAPSO algorithms described later in this thesis partially use the social-only model defined in Kennedy's study, i.e., it removes the original cognitive part of the velocity update equation. Besides, herein both algorithms alternate between the social-only model and the negative gradient direction.

2.6 Gradient descent

This section defines the gradient vector and presents a simple formulation of the gradient descent method for optimization. These definitions are used in the proposed SAPSO, ISAPSO, and DGHPSOGS algorithms.

Conventional methods of optimization employ tools such as gradient, subgradient and second derivative information with Hessians. These methods can not yield more than local solutions. They are not capable of locating or identifying a global optimum (HORST; TUY, 1996). To find a local minimum, the gradient descent method iteratively takes small steps in the direction of the negative gradient, following

$$\vec{\mathbf{x}}^{t+1} = \vec{\mathbf{x}}^{t} - \gamma \nabla f(\vec{\mathbf{x}}^{t}), \tag{2.15}$$

where $\vec{\mathbf{x}}^{t}$ is a multidimensional point approximated to the local minimum at iteration $t \geq 0$, $\gamma \in (0, 1)$ is the step size, and $\nabla f(\vec{\mathbf{x}}^{t})$ is the gradient of the multivariable function $f(\cdot)$ evaluated at $\vec{\mathbf{x}}^{t}$. The gradient is a vector corresponding to the partial derivatives of $f(\cdot)$ is given by

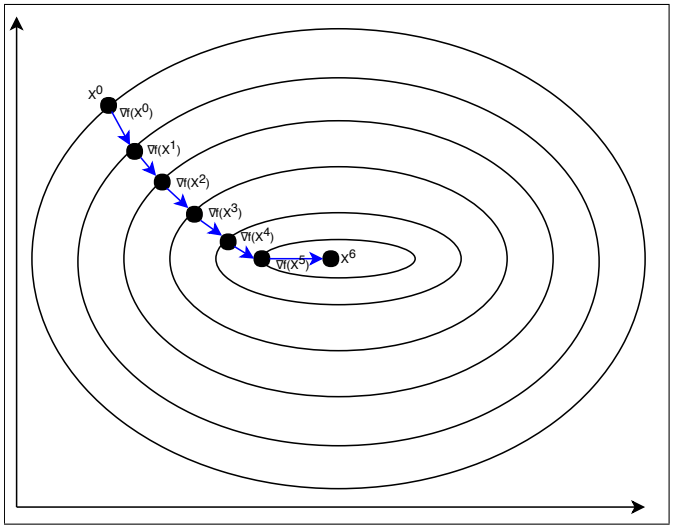
$$\nabla f = \frac{\partial f}{\partial x_{i1}^{t}} \vec{\mathbf{e}}_1 + \frac{\partial f}{\partial x_{i2}^{t}} \vec{\mathbf{e}}_2 + \cdots + \frac{\partial f}{\partial x_{id}^{t}} \vec{\mathbf{e}}_d, \tag{2.16}$$

where $\vec{\mathbf{e}}_{i=1,2,\ldots,d}$ are the orthogonal unit vectors pointing in the coordinate directions, which must be defined and differentiable at $\vec{\mathbf{x}}^{t}$. The partial derivatives can be approximated by a forward-differencing at a given dimension x_{ij}^{t} for a small number ε denote as

$$\frac{\partial f}{\partial x_{i}^{t}} \approx \frac{f(x_{i}^{t} + \varepsilon) - f(x_{i}^{t})}{\varepsilon}. \tag{2.17}$$

The gradient scheme expects to reach $f(\vec{\mathbf{x}}^{0}) \geq f(\vec{\mathbf{x}}^{1}) \geq \ldots \geq f(\vec{\mathbf{x}}^{t})$ at each iteration, where $\vec{\mathbf{x}}^{t}$ corresponds to a point at a non-prohibitive computing iteration number t where the desired minimum is found. Figure 4 illustrates the gradient descent algorithm starting at the point $\vec{\mathbf{x}}^{0}$ and finishing at the point $\vec{\mathbf{x}}^{6}$, considering the smallest level curve as the

Figure 4 – Illustration of the gradient descent in series of level curves (contour lines).



Source: produced by the author.

minimum value of the function. The common stop criterion is to assume a very small error ξ, such that $f(\vec{\mathbf{x}}^t) - f(\vec{\mathbf{x}}_{opt}) \leq \xi$, where $\vec{\mathbf{x}}_{opt}$ is the known *global minimum*. When the global minimum is not known, T can be used as the maximum iteration number of the algorithm.

2.7 Newton's and Quasi-Newton methods

The Newton's method, also known as Newton-Raphson (HANSEN, 1986), is used to find approximated roots of real-valued functions. Newton's method is an iterative approach that continuously looks for the point $(\vec{\mathbf{x}}, f(\vec{\mathbf{x}}) = 0)$ of a differentiable function f. With an initial guess $\vec{\mathbf{x}}^0$ and the derivative f', the following equation satisfies the better approximation of the next point $\vec{\mathbf{x}}^1$ given by

$$\vec{\mathbf{x}}^{t+1} = \vec{\mathbf{x}}^t - \frac{f(\vec{\mathbf{x}}^t)}{f'(\vec{\mathbf{x}}^t)}. \tag{2.18}$$

However, a common application of the method is in the optimization area, by applying the Newton's method in the differentiable function f' instead of the original function f. The method construct a sequence of points from an initial guess that converges towards an local optimum, i.e., find roots of a derivative function f', known as stationary points in the form of $(\vec{\mathbf{x}}, f'(\vec{\mathbf{x}}) = 0)$. The function must be twice-differentiable and the iterative method is repeated according to

$$\vec{\mathbf{x}}^{t+1} = \vec{\mathbf{x}}^t - \frac{f'(\vec{\mathbf{x}}^t)}{f''(\vec{\mathbf{x}}^t)}. \tag{2.19}$$

In higher dimensions, the derivative of the functions is replaced by the gradient $\nabla f(\vec{\mathbf{x}})$ and the second-order derivative is substituted by the inverse of the Hessian matrix

$\mathbf{H}f(\vec{\mathbf{x}})$ described as

$$\vec{\mathbf{x}}^{t+1} = \vec{\mathbf{x}}^{t} - \gamma[\mathbf{H}f(\vec{\mathbf{x}}^{t})]^{-1}\nabla f(\vec{\mathbf{x}}^{t}), \tag{2.20}$$

where $\gamma \in (0, 1)$ is the step size and $[\mathbf{H}f(\vec{\mathbf{x}}^{t})]^{-1}$ is the inverse of the Hessian matrix. A Hessian is a square matrix of second-order partial derivatives defined as follows

$$\mathbf{H} = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_d} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_d} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_d \partial x_1} & \frac{\partial^2 f}{\partial x_d \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_d^2} \end{bmatrix}.$$

One can note that the estimation of the Hessian matrix is computationally expensive, as the number of dimensions of a function increases. Due to this shortcoming, there are many quasi-Newton methods, where an approximation for the Hessian (or its inverse) $\mathbf{B}^t$ is built up from changes in the gradient. In quasi-Newton methods the Hessian matrix of second derivatives is not computed. The Hessian is updated by analyzing successive gradient vectors instead. A well-known quasi-Newton method is the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm (FLETCHER, 1987), which is the approach used by the GPSO algorithm to compute a local optimization around the global best position of the swarm. The approximated Hessian matrix is compute according to

$$\mathbf{B}^{t+1} = \mathbf{B}^{t} + \frac{\vec{\mathbf{y}}^{t}[\vec{\mathbf{y}}^{t}]^{T}}{[\vec{\mathbf{y}}^{t}]^{T}\vec{\mathbf{s}}^{t}} - \frac{\mathbf{B}^{t}\vec{\mathbf{s}}^{t}[\vec{\mathbf{s}}^{t}]^{T}\mathbf{B}^{t}}{[\vec{\mathbf{s}}^{t}]^{T}\mathbf{B}^{t}\vec{\mathbf{s}}^{t}}, \tag{2.21}$$

where $\mathbf{y}^{t} = \nabla f(\vec{\mathbf{x}}^{t+1}) - \nabla f(\vec{\mathbf{x}}^{t})$ and $\vec{\mathbf{s}}^{t} = \vec{\mathbf{x}}^{t+1} - \vec{\mathbf{x}}^{t}$. The first Hessian matrix $\mathbf{B}^{0}$ is usually the identity matrix. Thus, the first iteration of BFGS algorithm is the same one iteration of the gradient descent algorithm. Alternatively, an estimation of the inverse approximated Hessian matrix can be directly computed by

$$[\mathbf{B}^{t+1}]^{-1} = \left(\mathbf{I} - \frac{\vec{\mathbf{s}}^{t}[\vec{\mathbf{y}}^{t}]^{T}}{[\vec{\mathbf{y}}^{t}]^{T}\vec{\mathbf{s}}^{t}}\right)[\mathbf{B}^{t}]^{-1}\left(\mathbf{I} - \frac{\vec{\mathbf{y}}^{t}[\vec{\mathbf{s}}^{t}]^{T}}{[\vec{\mathbf{y}}^{t}]^{T}\vec{\mathbf{s}}^{t}}\right) + \frac{\vec{\mathbf{s}}^{t}[\vec{\mathbf{s}}^{t}]^{T}}{[\vec{\mathbf{y}}^{t}]^{T}\vec{\mathbf{s}}^{t}}, \tag{2.22}$$

where $\mathbf{I}$ is the identity matrix with the same order as $\mathbf{B}$.

Substituting $[\mathbf{H}f(\vec{\mathbf{x}}^{t})]^{-1}$ (Equation 2.20) by $[\mathbf{B}^{t}]^{-1}$, the following iterative BFGS approach is finally obtained

$$\vec{\mathbf{x}}^{t+1} = \vec{\mathbf{x}}^{t} - \gamma[\mathbf{B}^{t}]^{-1}\nabla f(\vec{\mathbf{x}}^{t}). \tag{2.23}$$

2.8 Scale, translation and rotation (in)variance

One important property of any optimization algorithm is the dependence (or not) of the rotation in which an optimization problem is defined. In (WILKE; KOK; GROENWOLD, 2007b), the authors described a mathematical tool to prove whether

an algorithm is rotationally invariant under scale, rotation and translation. Herein, one can highlight only the *rotation variance* property in which the search performance is dependent on the coordinate system of the objective function. Such property is related to the *separability* of the objective function, i.e., the optimal value for the ith coordinate does not depend on the choice of the remaining coordinates. Many well-known test functions are additively decomposable. They can be written as a sum of d one-dimensional functions $f(\vec{\mathbf{x}}) = \sum_{i=1}^{d} f_i(x_i)$. Additively decomposable functions are separable while separable functions are not necessarily additively decomposable (HANSEN et al., 2011).

Let a relation between two multidimensional points $\hat{\vec{\mathbf{x}}}$ and $\vec{\mathbf{x}}$ be through a scale factor s, translation by a vector $\vec{\mathbf{t}}$, and rotation by a proper orthogonal matrix Q, such that

$$\hat{\vec{\mathbf{x}}} = sQ\vec{\mathbf{x}} + \vec{\mathbf{t}} \tag{2.24}$$

for any $s \in \mathbb{R}$, $Q \in \mathrm{Orth}^+$, and $\vec{\mathbf{t}} \in \mathbb{R}^d$. The same function is expressed in each of these reference frames as

$$f(\hat{\vec{\mathbf{x}}}) = f(sQ\vec{\mathbf{x}} + \vec{\mathbf{t}}). \tag{2.25}$$

An alternative but equivalent interpretation is defined as two distinct functions $f(\cdot)$ and $\hat{f}(\cdot)$. In this case, by definition

$$f(\vec{\mathbf{x}}) = \hat{f}(\hat{\vec{\mathbf{x}}}), \tag{2.26}$$

where $\vec{\mathbf{x}}$ and $\hat{\vec{\mathbf{x}}}$ represent two design vectors. Substituting Equation 2.24 into Equation 2.26 gives

$$f(\vec{\mathbf{x}}) = \hat{f}(sQ\vec{\mathbf{x}} + \vec{\mathbf{t}}). \tag{2.27}$$

Let an additional vector transformation rule be $\vec{\mathbf{v}} = \vec{\mathbf{x}}_1 - \vec{\mathbf{x}}_2$. In addition, the corresponding vectors $\hat{\vec{\mathbf{x}}}_1$ and $\hat{\vec{\mathbf{x}}}_2$ be also $\hat{\vec{\mathbf{v}}} = \hat{\vec{\mathbf{x}}}_1 - \hat{\vec{\mathbf{x}}}_2$. Then, the relation between $\vec{\mathbf{v}}$ and $\hat{\vec{\mathbf{v}}}$ is

$$\begin{aligned} \hat{\vec{\mathbf{v}}} &= \hat{\vec{\mathbf{x}}}_1 - \hat{\vec{\mathbf{x}}}_2 \\ &= \left(sQ\vec{\mathbf{x}}_1 + \vec{\mathbf{t}}\right) - \left(sQ\vec{\mathbf{x}}_2 + \vec{\mathbf{t}}\right) \\ &= sQ\left(\vec{\mathbf{x}}_1 - \vec{\mathbf{x}}_1\right) \\ &= sQ\vec{\mathbf{v}} \end{aligned} \tag{2.28}$$

for any $s \in \mathbb{R}$ and $Q \in \mathrm{Orth}^+$. Any optimization algorithm is scale, translation and rotation invariant *if and only if* the transformation rules given by Equations 2.24 and 2.28 are satisfied for all $Q \in \mathrm{Orth}^+$, i.e., the same optimal results are then obtained irrespective of the scale and reference frame used.

When an optimization algorithm is scale, rotation and/or translation invariance means that the algorithm performance is independent of uniform scaling of all variables, rotation and/or translation of the reference optimization problem. When a particular optimization algorithm is frame dependent, there exists a specific choice of reference frame

in which the problem can be solved easier with less iterations or better by achieving a lower cost function value, as compared to some other reference frame (WILKE; KOK; GROENWOLD, 2007b). Since a priori knowledge of the optimal reference frame for a particular problem is seldom available, this places an additional burden on the analyst, which now has to consider solving the problem in a number of reference frames.

This implies that frame-dependent optimization algorithms are specialist algorithms, tuned to perform well on a special subclass of problems. In contrast, frame invariant optimization algorithms are general algorithms, applicable to a larger class of problems. Frame invariance of optimization algorithms is not a strict requirement. Frame invariance provides a useful classification of optimization algorithms. A frame invariant algorithm requires less a priori knowledge (or tacit assumptions) of the optimization problem, when compared to a frame variant algorithm. This implies that a frame invariant algorithm will have either inferior or superior performance if compared to a frame variant algorithm, depending on the validity of the assumed information.

Even though frame invariance is not required, some classes of optimization algorithms do satisfy this principle, such as gradient based optimization (SNYMAN, 2005). The gradient vector of any scalar function satisfies the transformation rules for changing both scale and reference frame. This renders classical gradient-based optimization algorithms scale and reference frame invariant. The following two subsections describes how the above mathematical formulations relates with PSO.

2.8.1 Rotationally invariant PSO version

PSO described in Section 2.1 is recalled here by the two main particle's formulations which carries out the movement of the candidate solution throughout the search space,

$$\vec{\mathbf{x}}_i^{t+1} = \vec{\mathbf{x}}_i^t + \vec{\mathbf{v}}_i^{t+1} \tag{2.29}$$

and

$$\vec{\mathbf{v}}_i^{t+1} = w\vec{\mathbf{v}}_i^t + \vec{\boldsymbol{\psi}}_i^t, \tag{2.30}$$

where $\vec{\boldsymbol{\psi}}_i^t$ is defined as *stochastic vector*.

The stochastic vector of the classical PSO version is given by

$$\vec{\boldsymbol{\psi}}_i^t = c_1\phi_1\left(\vec{\mathbf{p}}_i^t - \vec{\mathbf{x}}_i^t\right) + c_2\phi_2\left(\vec{\mathbf{g}}^t - \vec{\mathbf{x}}_i^t\right). \tag{2.31}$$

In order to validate this PSO version as rotationally invariant, and also scaled and translated invariant, the transformed stochastic vector $\hat{\vec{\boldsymbol{\psi}}}_i^t$ must hold the rules described in

Equations 2.24 and 2.28. The transformed stochastic vector $\hat{\vec{\boldsymbol{\psi}}}_i^t$ is given by

$$\begin{aligned}
\hat{\vec{\boldsymbol{\psi}}}_i^t &= c_1\phi_1\Big(\hat{\vec{\mathbf{p}}}_i^t - \hat{\vec{\mathbf{x}}}_i^t\Big) + c_2\phi_2\Big(\hat{\vec{\mathbf{g}}}^t - \hat{\vec{\mathbf{x}}}_i^t\Big) \\
&= c_1\phi_1\Big(sQ\vec{\mathbf{p}}_i^t + \vec{\mathbf{t}} - sQ\vec{\mathbf{x}}_i^t - \vec{\mathbf{t}}\Big) + c_2\phi_2\Big(sQ\vec{\mathbf{g}}^t + \vec{\mathbf{t}} - sQ\vec{\mathbf{x}}_i^t - \vec{\mathbf{t}}\Big) \\
&= sQc_1\phi_1\Big(\vec{\mathbf{p}}_i^t - \vec{\mathbf{x}}_i^t\Big) + sQc_2\phi_2\Big(\vec{\mathbf{g}}^t - \vec{\mathbf{x}}_i^t\Big) \\
&= sQ\Big(c_1\phi_1\Big(\vec{\mathbf{p}}_i^t - \vec{\mathbf{x}}_i^t\Big) + c_2\phi_2\Big(\vec{\mathbf{g}}^t - \vec{\mathbf{x}}_i^t\Big)\Big) \\
&= sQ\vec{\boldsymbol{\psi}}_i^t.
\end{aligned} \tag{2.32}$$

Equation 2.32 uses both Equations 2.24 and 2.31. The particle's position is defined as

$$\vec{\mathbf{x}}_i^{t+1} = \vec{\mathbf{x}}_i^t + w\vec{\mathbf{v}}_i^t + \vec{\boldsymbol{\psi}}_i^t \tag{2.33}$$

and for $\hat{\vec{\mathbf{x}}}_i^{t+1}$ is given by

$$\begin{aligned}
\hat{\vec{\mathbf{x}}}_i^{t+1} &= \hat{\vec{\mathbf{x}}}_i^t + w\hat{\vec{\mathbf{v}}}_i^t + \hat{\vec{\boldsymbol{\psi}}}_i^t \\
&= sQ\vec{\mathbf{x}}_i^t + \vec{\mathbf{t}} + sQw\vec{\mathbf{v}}_i^t + sQ\vec{\boldsymbol{\psi}}_i^t \\
&= sQ\Big(\vec{\mathbf{x}}_i^t + w\vec{\mathbf{v}}_i^t + \vec{\boldsymbol{\psi}}_i^t\Big) + \vec{\mathbf{t}} \\
&= sQ\vec{\mathbf{x}}_i^{t+1} + \vec{\mathbf{t}},
\end{aligned} \tag{2.34}$$

where now Equations 2.24 and 2.33 are used. The construction of the stochastic vector and position update rule of the classical velocity update equation given in Equations 2.32 and 2.34, respectively, satisfy the respective transformation rules described in Equations 2.28 and 2.24, $\forall Q \in \mathrm{Orth}^+$. Hence, the classical velocity update equation is scale, translation and rotationally invariant.

2.8.2 Rotationally variant PSO version

The stochastic velocity vector of another well-known PSO version is given by

$$\vec{\boldsymbol{\psi}}_i^t = c_1\boldsymbol{\Phi}_1\Big(\vec{\mathbf{p}}_i^t - \vec{\mathbf{x}}_i^t\Big) + c_2\boldsymbol{\Phi}_2\Big(\vec{\mathbf{g}}^t - \vec{\mathbf{x}}_i^t\Big), \tag{2.35}$$

where $\boldsymbol{\Phi}_{l=1,2}$ is a diagonal random matrix[1] uniformly distributed between $[0, 1]$. Similarly, the transformed stochastic velocity vector is derived by

$$\begin{aligned}
\hat{\vec{\boldsymbol{\psi}}}_i^t &= c_1\hat{\boldsymbol{\Phi}}_1\Big(\hat{\vec{\mathbf{p}}}_i^t - \hat{\vec{\mathbf{x}}}_i^t\Big) + c_2\hat{\boldsymbol{\Phi}}_2\Big(\hat{\vec{\mathbf{g}}}^t - \hat{\vec{\mathbf{x}}}_i^t\Big) \\
&= c_1\hat{\boldsymbol{\Phi}}_1\Big(sQ\vec{\mathbf{p}}_i^t + \vec{\mathbf{t}} - sQ\vec{\mathbf{x}}_i^t - \vec{\mathbf{t}}\Big) + c_2\hat{\boldsymbol{\Phi}}_2\Big(sQ\vec{\mathbf{g}}^t + \vec{\mathbf{t}} - sQ\vec{\mathbf{x}}_i^t - \vec{\mathbf{t}}\Big) \\
&= sc_1\hat{\boldsymbol{\Phi}}_1Q\Big(\vec{\mathbf{p}}_i^t - \vec{\mathbf{x}}_i^t\Big) + sc_2\hat{\boldsymbol{\Phi}}_2Q\Big(\vec{\mathbf{g}}^t - \vec{\mathbf{x}}_i^t\Big).
\end{aligned} \tag{2.36}$$

[1] Sometimes this PSO version has another notation where the diagonal random matrices are represented by a vector of random numbers.

Since $\hat{\vec{\boldsymbol{\psi}}}_i^t$ is the summation of two difference vectors, it follows from Equation 2.28 that the required transformation between $\hat{\vec{\boldsymbol{\psi}}}_i^t$ and $\vec{\boldsymbol{\psi}}_i^t$ is given by

$$\begin{aligned}\hat{\vec{\boldsymbol{\psi}}}_i^t &= sQ\vec{\boldsymbol{\psi}}_i^t \\ &= sQ\Big(c_1\boldsymbol{\Phi}_1\big(\vec{\mathbf{p}}_i^t - \vec{\mathbf{x}}_i^t\big) + c_2\boldsymbol{\Phi}_2\big(\vec{\mathbf{g}}^t - \vec{\mathbf{x}}_i^t\big)\Big) \\ &= sc_1Q\boldsymbol{\Phi}_1\big(\vec{\mathbf{p}}_i^t - \vec{\mathbf{x}}_i^t\big) + sc_2Q\boldsymbol{\Phi}_2\big(\vec{\mathbf{g}}^t - \vec{\mathbf{x}}_i^t\big).\end{aligned} \tag{2.37}$$

When comparing the transformation obtained in Equation 2.36 to the required transformation in Equation 2.37, satisfaction of scale and frame invariance requires

$$\begin{aligned}&\hat{\boldsymbol{\Phi}}_l Q = Q\boldsymbol{\Phi}_l \\ &\therefore \hat{\boldsymbol{\Phi}}_l = Q\boldsymbol{\Phi}_l Q^T, \quad \text{for } l = 1, 2 \text{ and } \forall Q \in \text{Orth}^+.\end{aligned} \tag{2.38}$$

Since the relation between $\hat{\boldsymbol{\Phi}}_l$ and $\boldsymbol{\Phi}_l$ has to hold $\forall Q \in \text{Orth}^+$, the solution for Equation 2.38 is

$$\hat{\boldsymbol{\Phi}}_l = \boldsymbol{\Phi}_l = a_l I \quad \text{for } l = 1, 2, \tag{2.39}$$

a_1 and a_2 are any real scalar values. If these values are uniform random scalars, this PSO version reduces to the rotationally invariant one. However, $\hat{\boldsymbol{\Phi}}_l$ and $\boldsymbol{\Phi}_l$ are random matrices generated independently. Thus, rotational variance is not satisfied. By scaling the components in the classical velocity update equation, both the vector magnitudes and vector directions are stochastically varied, which introduces both magnitude and directional diversity.

Accepting this PSO version as rotationally variant, let substitute $Q = I$ and assume that the random matrices $\hat{\boldsymbol{\Phi}}_l = \boldsymbol{\Phi}_l$ are generated to be equal for the remainder of the investigation to evaluate only the scale and translational variance. Following the previous Equation 2.33, the transformed position update equation for $\hat{\vec{\mathbf{x}}}_i^t$ is given by

$$\begin{aligned}\hat{\vec{\mathbf{x}}}_i^{t+1} &= \hat{\vec{\mathbf{x}}}_i^t + w\hat{\vec{\mathbf{v}}}_i^t + \hat{\vec{\boldsymbol{\psi}}}_i^t \\ &= s\vec{\mathbf{x}}_i^t + \vec{\mathbf{t}} + sw\vec{\mathbf{v}}_i^t + s\Big(c_1\boldsymbol{\Phi}_1\big(\vec{\mathbf{p}}_i^t - \vec{\mathbf{x}}_i^t\big) + c_2\boldsymbol{\Phi}_2\big(\vec{\mathbf{g}}^t - \vec{\mathbf{x}}_i^t\big)\Big) \\ &= s\Big(\vec{\mathbf{x}}_i^t + w\vec{\mathbf{v}}_i^t + c_1\boldsymbol{\Phi}_1\big(\vec{\mathbf{p}}_i^t - \vec{\mathbf{x}}_i^t\big) + c_2\boldsymbol{\Phi}_2\big(\vec{\mathbf{g}}^t - \vec{\mathbf{x}}_i^t\big)\Big) + \vec{\mathbf{t}} \\ &= s\vec{\mathbf{x}}_i^{t+1} + \vec{\mathbf{t}}.\end{aligned} \tag{2.40}$$

Equation 2.40 shows that this PSO version is strictly invariant of scale and translation, since it satisfies Equation 2.24 exactly, but is rotationally invariant as Equation 2.36 does not satisfy Equation 2.37 exactly.

This mathematical tool is used later in Section 3.3 to prove that ISAPSO algorithm is strictly rotationally invariant in the attraction phase of the algorithm and rotationally invariant in a stochastic sense when the swarm is in the repulsion phase.

2.9 Versions of particle swarm optimization algorithm

This section describes six algorithms related to the thesis: (Section 2.9.1) attractive and repulsive (ARPSO), which is a diversity-guided algorithm; (Section 2.9.2) gradient-based PSO (GPSO); (Section 2.9.3) diversity-guided hybrid PSO based on gradient search (DGHPSOGS); (Section 2.9.4) a standard PSO (SPSO); (Section 2.9.5) Wilke PSO (WPSO); and (Section 2.9.6) Bonyadi PSO (BPSO). This section also discusses about the functioning of each algorithm, highlighting their main advantages and drawbacks.

2.9.1 Attractive and repulsive PSO – ARPSO and ARPSO*

The classical PSO tends to suffer from premature convergence when applied to highly multimodal test functions. One of the reasons of this fact is due to the decrease of the swarm's diversity during the search procedure which leads to a fitness stagnation of the swarm. In (VESTERSTRØM; RIGET, 2002), the authors raise the hypothesis that maintenance of high diversity is crucial for preventing premature convergence in multimodal optimization. This hypothesis boosted the development of two algorithms: ARPSO and ARPSO*. Both versions of the algorithm differ in the stop criterion only. The ARPSO algorithm uses a diversity measure to alternate between exploring and exploiting behavior through two phases: *attraction* and *repulsion*. The measure is expressed as

$$diversity(\vec{\mathbf{x}}^t) = \frac{1}{n \times |L|} \times \sum_{i=1}^{n} \sqrt{\sum_{j=1}^{d} (x_{ij}^t - \overline{x}_j^t)^2}, \tag{2.41}$$

where $|L|$ is the maximum radius of the search space, x_{ij}^t is the j-th dimension of the ith particle at t iteration, and $\overline{x}_j^t$ is the j-th dimension of the average position over all particles. By measuring the diversity of the swarm, the algorithm alternates between those two phases according to

$$\vec{\mathbf{v}}_i^{t+1} = w^t \vec{\mathbf{v}}_i^t + dir \times \left[c_1 \dot{\phi}_i^t \left(\vec{\mathbf{p}}_i^t - \vec{\mathbf{x}}_i^t \right) + c_2 \ddot{\phi}_i^t \left(\vec{\mathbf{g}}^t - \vec{\mathbf{x}}_i^t \right) \right], \tag{2.42}$$

subject to

$$dir = \begin{cases} -1 & diversity < d_{low} \\ 1 & diversity > d_{high} \end{cases}. \tag{2.43}$$

In the attraction phase ($dir = 1$), the swarm is contracting and consequently the diversity value decreases. When the diversity value drops below a lower bound, d_{low}, the swarm switches to the repulsion phase ($dir = -1$). When the swarm reaches the upper bound, d_{high}, the swarm switches back to the attraction phase. The ARPSO algorithm alternates between both phases of attraction and repulsion during the whole search process.

Despite of attraction and repulsion mechanism of ARPSO algorithm, which adds the features of escaping from local minima and improves its search ability, the algorithm

suffers from the lack of fine-tuning feature. Eventually, the valley of the global minimum of a multimodal function will be reached, and the algorithm will be limited to exploit it until the diversity drops below d_{low} parameter. The feature of escaping from local minima with diversity control is probably the one that also weakens the fine-tuning. The stop criterion of ARPSO algorithm is defined as the maximum number of evaluations of the objective function which is 2000 times the number of dimensions of the function.

The ARPSO* algorithm defines the stop criterion to be 200000 evaluations without fitness improvements. This ARPSO* version results in better performance than the original ARPSO. However, the comparison between the two versions seems to be unfair, since more search time is given to the weak fine-tuning to improve the final result.

2.9.2 Gradient-based PSO – GPSO

Noel's work (NOEL, 2012) came up with an important discussion about deterministic and non-deterministic approaches in optimization problems. On the one hand, the stochastic optimization algorithms perform global search but waste computation effort by applying random search. On the other hand, deterministic algorithms converge rapidly but may get stuck in local minima of multimodal functions. Keeping this in mind, a hybrid optimization algorithm, referred as GPSO, was proposed. This algorithm combines the strengths of stochastic and deterministic approaches and tries to avoid their weaknesses. The GPSO algorithm uses the classical PSO with velocity update equation described by Equation 2.3, and gradient-based local search algorithm to achieve faster convergence and better accuracy of the final solution without getting trapped in local minima.

The GPSO algorithm avoids the use of inertia weights and constriction coefficients which may lead to a local minimum if improperly chosen. After each iteration of the algorithm, it is applied the BFGS method (see Section 2.7 for more details) under five iterations to perform local search. The initial guess of the Quasi-Newton algorithm is the global best position found by the PSO algorithm. The Quasi-Newton algorithm uses the second derivative (Hessian matrix) and makes use of information about the curvature in addition to the gradient information. The authors of the GPSO algorithm discovered that performing the local search periodically, i.e., after each PSO iteration instead of every ten PSO iterations, yielded better results for multimodal functions.

Although the GPSO algorithm can fine-tune the global best position at each iteration of PSO, there is no feature to escape from local minima if the global position is trapped in a valley of a multimodal function. The only mechanism of escaping of this condition is to hopefully wait for a particle to fly over another valley deeper than the current one where the global best particle is. Consequently, the position of this particle will become the new global best position and the fine-tune (with Quasi-Newton algorithm) will get this position as the starting point. In the worst case, the whole swarm will be in

one valley, giving no chance to escape from that pitfall and ending up with one suboptimal solution of the problem.

2.9.3 Diversity-guided hybrid PSO based on gradient search – DGHPSOGS

A diversity-guided hybrid PSO based on gradient search is proposed to improve the search ability of the swarm in (HAN; LIU, 2014). To obtain better performance, an adaptive PSO algorithm is used based on (SHI; EBERHART, 1998a), where the inertia weight is computed as a function of the number of iterations according to the Equation 2.7. The idea of attractive and repulsive schemes is also presented in this approach. The particle velocity update equation is expressed as

$$\vec{\mathbf{v}}_i^{t+1} = w^t \vec{\mathbf{v}}_i^t + c_1 \dot{\phi}_i^t \left(\frac{-\partial f / \partial \vec{\mathbf{x}}_i^t}{|| - \partial f / \partial \vec{\mathbf{x}}_i^t ||} \right) - c_2 \ddot{\phi}_i^t \left(\frac{\left(\vec{\mathbf{g}}^t - \vec{\mathbf{x}}_i^t \right)}{\text{diversity}(\vec{\mathbf{x}}^t) + \xi} \right), \tag{2.44}$$

where the second term corresponds to the normalized gradient direction related to the partial derivatives of $\vec{\mathbf{x}}_i^t$, and ξ in the third term is a predetermined small positive number, to ensure the denominator will never be equal to zero.

The proposed hybrid method combines random search and semi-deterministic search in one algorithm. First, an adaptive PSO (APSO) is used to perform random search while the diversity of the swarm is less than a predetermined threshold (d_{low}). The particle velocity update equation of APSO is defined in Equation 2.6 with time-varying inertia weight. When the swarm loses its diversity, the particles' velocity update equation switches to the Equation 2.44 and the algorithm performs a semi-deterministic search while the diversity is above the same threshold (d_{low}). The search alternates between Equations 2.6 and 2.44, based on the diversity value of the swarm. Equation 2.6 may cause the swarm to loose its diversity and get stuck in local minima with high likelihood, whereas Equation 2.44 can keep the diversity of the swarm adaptively as well as search in the negative gradient direction, which may increase the likelihood of finding a global minimum.

Despite the new idea proposed for hybridization, the parameters of DGHPSOGS seem to be very problem-dependent. The authors had to set up, for each test function, different values for the diversity threshold, cognitive and social coefficients to find interesting results. It is also worth noting the second and third terms of Equation 2.44. It seems contradictory to search in the negative gradient direction of each particle position (second term), i.e., in the direction of the local minimum, and repel the particles at the same time (third term). Also, only one threshold is defined, the lower bound d_{low}. Therefore, the process of repelling particles does not seem to last long enough to properly escape from local minima, once the diversity value quickly restores its level above the d_{low} threshold, forcing the swarm to use the APSO again, i.e., the particles tend to crowd as quickly as possible again.

2.9.4 Standard PSO – SPSO

This PSO version was developed by Maurice Clerc in (CLERC, 2012b) and later updated by Mauricio Zambrano-Bigiarini in (ZAMBRANO-BIGIARINI; CLERC; ROJAS-MUJICA, 2013). This version is known as Standard Particle Swarm Optimization (SPSO), which is an attempt to create a PSO with a velocity update equation modified in a geometrical way in which is not dependent of the coordinate system, since classical PSO version has biases in the direction of the axes (SPEARS; GREEN; SPEARS, 2010; MONSON; SEPPI, 2005).

Such PSO version generates a random point in a perfect hyper-spherical distribution formed by two arguments: i) center of gravity among the current particle's position, a random point between the particle's position and the local memory, and another random point between the particle's position and the global memory; and ii) a radius between the particle's position and the center of gravity. The center of gravity is denoted as $\vec{\mathbf{c}}_i^t$ and performed by

$$\vec{\mathbf{c}}_i^t = \frac{\vec{\mathbf{x}}_i^t + \vec{\mathbf{P}}_i^t + \vec{\mathbf{G}}_i^t}{3}, \tag{2.45}$$

where vectors $\vec{\mathbf{P}}_i^t$ and $\vec{\mathbf{G}}_i^t$ are computed, respectively, as

$$\vec{\mathbf{P}}_i^t = \vec{\mathbf{x}}_i^t + c_1 \dot{\vec{\boldsymbol{\phi}}}_i^t \odot \left(\vec{\mathbf{p}}_i^t - \vec{\mathbf{x}}_i^t\right) \tag{2.46}$$

and

$$\vec{\mathbf{G}}_i^t = \vec{\mathbf{x}}_i^t + c_2 \ddot{\vec{\boldsymbol{\phi}}}_i^t \odot \left(\vec{\mathbf{g}}^t - \vec{\mathbf{x}}_i^t\right). \tag{2.47}$$

The velocity update equation of this PSO version is defined as

$$\vec{\mathbf{v}}_i^{t+1} = w^t \vec{\mathbf{v}}_i^t + \mathcal{H}\left(\vec{\mathbf{c}}_i^t, ||\vec{\mathbf{c}}_i^t - \vec{\mathbf{x}}_i^t||\right) - \vec{\mathbf{x}}_i^t, \tag{2.48}$$

where function $\mathcal{H}(\cdot,\cdot)$ is performed according to the Algorithm 4. The search distribution of a particle is obtained in this case as a d-dimensional sphere, which is invariant by rotation around its center, i.e. the search distribution is not modified when the surface response of the function is rotated.

Algorithm 4 $\mathcal{H}(\cdot,\cdot)$ function to draw a random number in a hyperspherical shape.

1: **function** $\mathcal{H}(\vec{\mathbf{c}}, r)$
2: $\quad\vec{\mathbf{z}} \leftarrow \mathcal{N}(\mathbf{0}, \mathbf{I})$ ▷ draw sample from a standard normal distribution
3: $\quad\vec{\mathbf{z}}' \leftarrow \vec{\mathbf{z}}/||\vec{\mathbf{z}}||$ ▷ Normalize vector
4: $\quad a \leftarrow \mathcal{U}(0, r)$ ▷ draw sample from an uniform distribution
5: $\quad\vec{\mathbf{y}} = \vec{\mathbf{c}} + a * \vec{\mathbf{z}}'$
6: $\quad$**return** $\vec{\mathbf{y}}$
7: **end function**

2.9.5 Wilke PSO – WPSO

In (WILKE; KOK; GROENWOLD, 2007a; WILKE; KOK; GROENWOLD, 2007b), the authors investigated whether PSO is invariant of the scale, rotation and translation in which an objective function is posed. The paper showed that linear velocity update equation lacks directional diversity, resulting in particle trajectories that collapse to line searches. On the other hand, the classical velocity update equation maintains diverse particle trajectories. Additionally, the authors provided a formal mathematical proof to show that linear PSO is scale and frame invariant, whereas classical PSO lacks rotational invariance.

Beyond the analyses of scale, rotation and translation, the authors also proposed a new PSO version, herein termed as WPSO, to illustrate that diversity and invariance are not necessarily exclusive. The velocity update equation is rotationally invariant in a stochastic sense and at the same time directionally diverse. This is achieved through consistent small perturbation in the direction of the local $(\vec{\mathbf{p}} - \vec{\mathbf{x}})$ and global $(\vec{\mathbf{g}} - \vec{\mathbf{x}})$ memories of the swarm by multiplying the above vectors with an independent random rotation matrix $\mathbf{W}$.

It is important to mention that building a proper orthogonal matrix $\mathbf{W}$ is usually computationally expensive, since several matrix-by-matrix multiplications are required. WPSO algorithm uses the exponential map through a simple series method (MOLER; LOAN, 2003). The general series expansion of an exponential map is given by

$$\mathbf{W}_i^t = I + \sum_i^k \frac{1}{i!}\left(\frac{\alpha\pi}{180}\left(\mathbf{A} - \mathbf{A}^T\right)\right)^i, \tag{2.49}$$

where $\mathbf{I}$ is the identity matrix, $\mathbf{A}$ is a $d \times d$ random matrix with each entry $\mathbf{A}_{ij} \sim \mathcal{U}(-0.5, 0.5)$, and α is a real scaling factor representing the perturbation against the directions of each memory. Finally, the velocity update equation is formulated according to

$$\vec{\mathbf{v}}_i^{t+1} = w^t\vec{\mathbf{v}}_i^t + c_1\dot{\phi}_i^t\dot{\mathbf{W}}_i^t\left(\vec{\mathbf{p}}_i^t - \vec{\mathbf{x}}_i^t\right) + c_2\ddot{\phi}_i^t\ddot{\mathbf{W}}_i^t\left(\vec{\mathbf{g}}^t - \vec{\mathbf{x}}_i^t\right). \tag{2.50}$$

As a result, the velocity update equation is strictly scale and translation invariant, and rotationally invariant in a stochastic sense. The authors demonstrated that the WPSO may outperform the classical PSO for test functions with a single local minimum. In addition, the performance of the WPSO is comparable to the performance of the classical PSO for the multimodal test functions, with the added advantage of being invariant of the reference frame.

2.9.6 Bonyadi PSO – BPSO

This paper examines several issues associated with PSO algorithm (BONYADI; MICHALEWICZ; LI, 2014; BONYADI; MICHALEWICZ, 2014). The main issues outlined

are: i) stagnation of particles in some points in the search space; ii) inability to change the value of one or more decision variables; iii) poor performance when the swarm size is small; iv) lack of guarantee to converge even to a local optimum (local optimizer); v) poor performance when the number of dimensions grows; vi) and sensitivity of the algorithm to the rotation of the search space. As an alternative to overcome these issues, a new general form of velocity update equation for the PSO algorithm is proposed.

This variation is based on WPSO (WILKE; KOK; GROENWOLD, 2007b) in which the perturbation of the movement vector contains two parts: magnitude perturbation and direction perturbation. The magnitude perturbation diversifies the step size of particles movement (WILKE; KOK; GROENWOLD, 2007a), while the direction perturbation guarantees the direction diversity in the swarm. These perturbations are important for the exploitation and exploration ability of the algorithm as was studied by (WILKE; KOK; GROENWOLD, 2007a). Note that if a small magnitude perturbation with large direction perturbation is applied to a particle, then the particle behaves more exploitatively. However, if the magnitude perturbation is large, then the particle behaves more exploratively. The new algorithm is called BPSO. The authors showed that this version of PSO with its new form of the velocity update equation addresses all primary issues (stagnation, local convergence and rotation variance) at the same time under some identified conditions. Equation 2.51 shows the velocity update equation of BPSO for the proposed specific model,

$$\vec{\mathbf{v}}_i^{t+1} = w^t \vec{\mathbf{v}}_i^t + c_1 \dot{\phi}_i^t \left(\mathcal{N}\left(\vec{\mathbf{p}}_i^t, \dot{\sigma} I\right) - \vec{\mathbf{x}}_i^t \right) + c_2 \ddot{\phi}_i^t \left(\mathcal{N}\left(\vec{\mathbf{g}}^t, \ddot{\sigma} I\right) - \vec{\mathbf{x}}_i^t \right), \tag{2.51}$$

where $\dot{\sigma}$ and $\ddot{\sigma}$ are the variances of the normal distribution $\mathcal{N}(\vec{\mathbf{p}}, \dot{\sigma} I)$ and $\mathcal{N}(\vec{\mathbf{g}}, \ddot{\sigma} I)$, respectively. In order to perform exploration in the earlier stages and exploitation in the later stages, larger values of variance at the beginning of the optimization and smaller values at the later stages are preferable. In this way, the authors proposed a dynamic variance during the search process in the form of $\sigma_{kt} = h_{kt}(\cdot, \cdot)$ for $k = 1, 2$. In this case, the normal distributions become $\mathcal{N}(\cdot, h_{kt}(\cdot, \cdot) I)$. There might be many different ways to design the function h_{kt}, and each may have a different impact on the performance of the algorithm. The authors preferred to use a function based on the Euclidean distance between $\vec{\mathbf{x}}$ and $\vec{\mathbf{p}}$ or $\vec{\mathbf{x}}$ and $\vec{\mathbf{g}}$ for h_{1t} and h_{2t}. The function $h_{1t} = h_{2t} = h_t$ is defined as

$$D_t = h_t(\vec{\mathbf{y}}, \vec{\mathbf{z}}) = \begin{cases} l D_{t-1} & \text{if } y_i = z_i \text{ for all } i, \\ l \sqrt{\sum_{i=1}^{d} \left(y_i - z_i\right)^2} & \text{otherwise,} \end{cases} \tag{2.52}$$

where $\vec{\mathbf{y}}$ and $\vec{\mathbf{z}}$ are two input vectors and $l > 0$ is a constant. Note that whenever the two inputs of the functions $h_{\vec{\mathbf{y}}, \vec{\mathbf{z}}}$ are equal, the previous calculated distance is used instead.

2.10 Clerc's rules

In (CLERC, 2012a), Maurice Clerc discusses about the prudence a practitioner must have when comparing two stochastic algorithms. The author proposed five rules to make the comparison fair enough between two metaheuristics. The rules are stated as follows:

- **Rule 1:** *Never trust biased problems for comparisons.*
- **Rule 2:** *Be sure that the estimated performance has converged reasonably.*
- **Rule 3a:** *Never trust just one performance criterion.*
- **Rule 3b:** *Always perform a statistical test.*
- **Rule 4:** *For reproducible results, do specify the word size.*
- **Rule 5:** *For reproducible results, do specify the Random Number Generator (RNG) that is used, including its parameters, if any (like the seeds).*

Rule 1 states the problem is biased when its solution is in one of the following positions: center, axis, or diagonal. The works (JANSON; MIDDENDORF, 2007; SPEARS; GREEN; SPEARS, 2010) explain this phenomenon whose PSO algorithm wrongly admits the solution is in the middle of the search space. Clerc's work (CLERC, 2012a) also emphasizes the higher density of particles near the center of the search space. Such phenomenon is also presented in GA (SALOMON, 1996), which may indicate that to reliably evaluate algorithms the test functions must be at least shifted from the center.

Rule 2 affirms that a small number of executions can lead to misleading results. An arbitrary maximum number of executions t_1 might indicate that an algorithm **A** is better than an algorithm **B**. However, another arbitrary number of executions t_2, such that $t_1 \ll t_2$, points to algorithm **B** better than **A**. This indecisiveness about the best of two algorithms regarding the number of runs could be solved with the Law of Large Numbers (LLN) (DURRETT, 2010). LLN says that the average of t independent trials tends to the expected value as $t \to \infty$. The weak LLN admits an error $\epsilon > 0$, thus the probability that the average value and the expected value differ by more than ϵ tends to 0 as $t \to \infty$. Let $S = e_1 + e_2 + \ldots, e_t$ be the sum of independent trial process e_i with continuous density function f, finite expected value μ, and finite variance σ^2. Then, for any real number $\epsilon > 0$, the probability must satisfy

$$\lim_{t \to \infty} P\left(\left|\frac{S}{t} - \mu\right| \geq \epsilon\right) = 0. \tag{2.53}$$

The Chebyshev's Inequality states that

$$P\left(\left|\frac{S}{t}-\mu\right|\geq\epsilon\right)\leq\frac{\sigma^2}{t\epsilon^2}. \tag{2.54}$$

However, the practitioner must have the expected value (μ) and variance (σ^2) along with the required error (ϵ) to accomplish the reliable number of experiments t. Obviously, in the context of optimization tasks, μ and σ^2 are not defined before the algorithm's run. Thus, LLN becomes impractical. Although Clerc states that the number of runs t is problem-dependent, a method to find a reasonable value for t is provided later in Section 3.2.3.

Rule 3a and 3b argue the relative independence among the performance measures (criteria). Let criteria ϕ_1 and ϕ_2 be the measures used to evaluate algorithms **A** and **B**. The algorithm **A** seems better than algorithm **B** with respect to the criterion ϕ_1, but algorithm **A** performs worse than the algorithm **B** with respect to criterion ϕ_2. The above scenario contributes to the statement that Rule 3a must be true at least for a small number of executions. Similarly, Rule 3b avoids the misunderstanding of an algorithm **A** (or **B**) be better than algorithm **B** (or **A**) by chance. To comply with both rules, two common criteria are used to evaluate the algorithms: average error rate and average success rate. In addition, two non-parametric hypothesis tests, named Friedman's and Wilcoxon's tests, are applied to evaluate the statistical significance of results.

Rule 4 is related to the numbers internally coded in the computer as words of b bits, assuming 2^b different values. All algorithms developed here are coded in MATLAB® 2018a whose floating-point numbers with double-precision data types according to IEEE® Standard 754 for double precision (INSTITUTE; ELECTRICAL; ENGINEERS, 1985; INSTITUTE; ELECTRICAL; ENGINEERS, 2008) are used. Therefore, values stored as *double* require 64 bits.

Rule 5 concerns about the Pseudo-Random Number Generator (PRNG). Some experimental tests derived from the Clerc's work showed that there is no relation between the quality of the randomness and the performance of the algorithm. In other words, one may have good performance with a bad PRNG, e.g., 3 bits per word, but on the other hand the performance may increase with a better PRNG, like 32 bits per word. In this paper, all codes use the 64-bit version of Mersenne Twister (MT19937-64) (MATSUMOTO; NISHIMURA, 1998) and all random numbers are drawn from a uniform distribution $\mathcal{U}(0,1)$, unless otherwise stated.

2.11 General formulation of optimization problems

Before proceeding with the proposed enhancements of this thesis, this section poses a general optimization problem by considering a two dimensional arbitrary function,

without loss of generality for higher dimensions, as many computing and engineering applications lie in optimizing an ordinary function. One can assume that the optimization problem is to find a global minimum, i.e., the function f to be minimized, since maximizing f is equivalent to minimizing $-f$. This statement is true for all numerical simulations found later in Chapter 4 by considering both benchmark suite: De Jong and CEC 2017.

The search space of a problem is large and highly dimensional, therefore any known optimization technique faces difficult challenges to reach feasible results. A general unconstrained optimization problem in a d-dimensional space can be defined as

$$\begin{aligned} &\text{Given} \quad f : \mathbb{R}^d \to \mathbb{R} \\ &\text{Find} \quad \vec{\mathbf{x}}_{opt} \mid f(\vec{\mathbf{x}}_{opt}) \leq f(\vec{\mathbf{x}}), \forall \vec{\mathbf{x}} \in \mathbb{R}^d, \end{aligned} \tag{2.55}$$

where f is an objective function that is to be minimized, $\vec{\mathbf{x}}$ is a cadidate solution represented by a vector, i.e., a point in $\mathbb{R}^d$ dimensional space, and $\vec{\mathbf{x}}_{opt}$ is the optimal solution known as global minimum. When f is highly multimodal, any classical deterministic approaches such as gradient descent and Newton's method can easily fall into one of the local minima, depending on the initial guess (HORST; TUY, 1996). Beyond that, some optimization problems have no closed form of f. Due to the lack of a functional representation, standard deterministic approaches such as *find* $\vec{\mathbf{x}} \mid \nabla f(\vec{\mathbf{x}}) = 0$ are not applicable, then the minimization of the function is not straightforward.

Particle swarm optimization algorithms work with and without the functional representation of the objective function f. Therefore, they belong to a class of *metaheuristic* optimization approaches and are suited for black box optimization. As optimization tasks are NP-hard problem, they intrinsically call for stochastic search techniques which can find a solution at least near to the global optimum.

In order to deepen the problematization of this thesis, the following equation

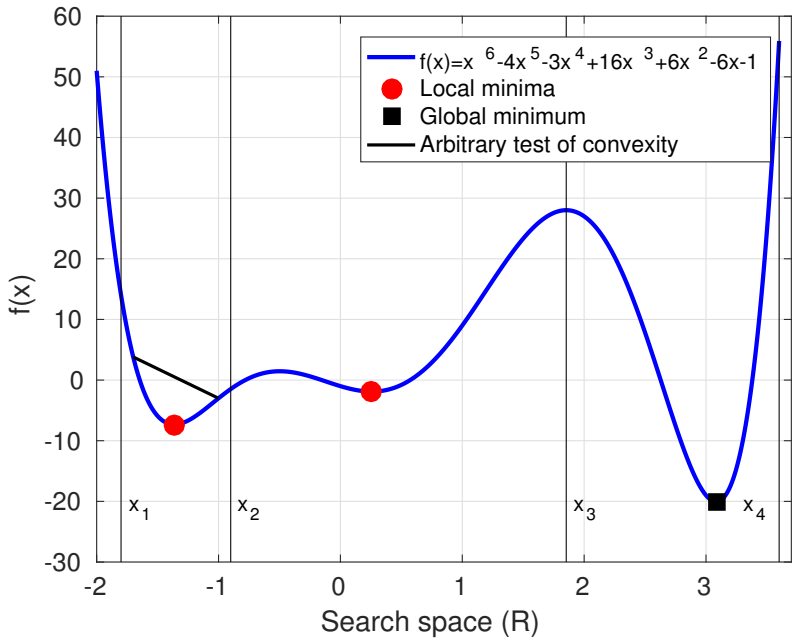
$$f(x) = x^6 - 4x^5 - 3x^4 + 16x^3 + 6x^2 - 6x - 1, \tag{2.56}$$

a one-dimensional six degree polynomial function with real roots, is yielded as a toy example to explain important topics. This polynomial function is multimodal and not convex. The multimodal functions are those with multiple local optima. A local optimum is a solution which is better than all its immediate neighbor solutions.

The polynomial function depicted in Figure 5 was constrained in the search domain $x \in [-2, 3.6]$ for simplicity in viewing. One can note three minima in the constrained search space $\mathbb{R}$. Note that two out of three minima are local optima, i.e., there are no better solutions around them. This polynomial function has only one global optimum, which is the best solution among all local minima.

A function is said convex if a straight line segment between any two points on the graph of the function lies above or on the graph. For instance, the polynomial function

Figure 5 – Six degree polynomial function defined in Equation 2.56.



Source: produced by the author.

presented in Figure 5 is not convex considering the whole search space. However, if the domain is constrained, such that $\mathbb{S} = \{x \mid x_1 \leq x \leq x_2\} \subset \mathbb{R}$, it becomes easy to show that:

$$\forall x_1', x_2' \in \mathbb{S}, \forall t \in [0,1]: \ f(tx_1' + (1-t)x_2') \leq tf(x_1') + (1-t)f(x_2'),$$

and the function is strictly convex if $x_1' \neq x_2'$ and $t \in (0,1)$. For visualization sake, the Figure 5 shows an arbitrary test of convexity with a straight line between $[x_1, x_2]$.

A deterministic optimization algorithm such as gradient descent or Newton's method would certainly find a local minimum (or global minimum) with the best performance. This function concedes the first- and second-order derivatives which would indicate the rate of change of the function at each iteration of the deterministic algorithm. These features also indicate the correct side to make the movement in the search domain, considering the initial guess at x^0, where the gradient scheme expects to reach $f(x^0) \geq f(x^1) \geq \ldots \geq f(x^t)$, where x^t corresponds to a point at a non-prohibitive computing iteration number t where the desired minimum is found.

The fundamental drawback of those deterministic approaches is that the practitioner must set the initial guess. If the quality of the initial guess is not good enough, i.e., the initial guess is not in the correct basin of attraction (the basin where the global optimum actually is), for instance between $[x_1, x_2]$ in the Figure 5, then the deterministic algorithm will converge to a local minima with no chances to escape. SAPSO and ISAPSO algorithms embed those deterministic features by using the negative gradient at a candidate solution point of a function. This information reduces the effect of random walk of each particle in

the swarm. In addition, the diversity control of the swarm can avoid local optima as a final result. Once the clustering of particles is detected, a trigger takes place and makes the swarm disperse throughout the search space. The mechanism of attraction and repulsion of particles is the one related to the trade-off between exploration and exploitation of the space. This mechanism is explained in detail in Chapter 3.

3 Enhancements for PSO: SAPSO, rotation and information exchange, and ISAPSO

This chapter outlines a detailed explanation of all contributions supplied by this thesis. The organization of this chapter is divided in chronological order of three parts: i) first the SAPSO algorithm is presented (SANTOS et al., 2018), ii) a rotation and information exchange is discussed, and iii) later the ISAPSO algorithm is reported as a final outcome of this thesis. Herein, the SAPSO algorithm is considered the starting point for enhancements in particle swarm. The current chapter also dedicates much attention to the exploration versus exploitation trade-off. This discussion is of interest to fully understand the SAPSO algorithm. Beyond that, the rotation and information exchange bring back the foundations of PSO algorithms whose performance is the essential motivation of this analysis. A method to define the number of executions is also explained in this section. This method selects a reliable number of experiments based on a set of benchmark functions, although it also works with only one objective function. This chapter ends with the description about ISAPSO algorithm, which is an enhancement proposed for SAPSO algorithm. Besides, a mathematical proof of the property *rotation invariance* based on Section 2.8 is formulated.

3.1 SAPSO: semi-autonomous particle swarm optimizer

The main contributtion of this method is to alternately use gradient information and stochastic search based on the current global best position found so far by the entire population to guide the search towards the global minimum. Both directions are used individually and independently by each particle of the swarm. Some particles will be flying in the direction of the global best position while others will be investigating their own neighborhood area. Therefore, the particles are referred as *autonomous*. Similarly, the SAPSO algorithm can be classified as an individual memetic algorithm (MOSCATO, 1989; HART; KRASNOGOR; SMITH, 2005), since each particle alternates between global and local search. Although an external decision taken by the swarm plays role in the final direction of the particle, which is discussed next.

Even though the particles can take individual decisions at different time during the search process, they are under a general rule of attraction and repulsion controlled by the swarm's diversity. Thus, the final direction of a particle is not fully determined by its own directive. In fact, the word *semi* arises due to this partial control of movement that a particle has, which is only completely defined with the swarm's choice between the phase

of attraction or repulsion. In summary, it is worth noting that the two important choices are taken at each iteration of SAPSO algorithm as follows:

1. The particle's choice to follow the gradient information of its position in the search space or to follow a stochastic search based on the global best position found so far by the swarm. This choice depends on the particle's current position in the search space. If the particle assumes that it is in a local minimum (a method to detect local minima is later described in Section 3.1.3), then it chooses to follow in the direction of the global best position, otherwise it chooses the gradient direction.

2. The swarm's choice to be in the attraction or repulsion phase. This choice depends on the diversity value of the swarm at each iteration of the algorithm. A high diversity value places the swarm in the attraction phase. The repulsion phase is used when the diversity value is low (see Section 2.9.1 to review the diversity controlling mechanism).

The mechanism of the SAPSO algorithm brings two strong features to avoid local minima during the search process: (i) each particle of the swarm has the ability to escape from local minima through its own decision of following the global tendency of the swarm after locally search an area of the space; (ii) the entire swarm can assume a repulsion scheme for a period of time and it avoids a suboptimal solution to the optimization problem.

Another interesting feature is that in the first iterations of the SAPSO algorithm, each particle is properly investigating its neighborhood through the gradient information, reducing the random walks influence and optimizing the search process. It is also important to highlight the SAPSO algorithm determines a sort of "individual exploitation", but it also maintains the traditional exploration regarding the scattered particles at the beginning of the execution. In the following subsection, a self-contained example of the algorithm is explained.

3.1.1 A short self-contained example of the SAPSO algorithm

As an effort to elucidate the online execution of the SAPSO algorithm, Figure 6 depicts important steps of the approach. The steps do not represent iterations in a sequence. In fact, they represent important snapshot moments of three particles (circle, square and diamond) in the search space that one wants to discuss. The order of movement is: circle first, square second, and diamond third. The numbers near the valleys are indicative of their depth, the higher the number the deeper the valley. The representative function has nine valleys, but only five of them are highlighted, which are the most important ones for the illustration. Note that the valley positioned at the center, which has the number 5

associated with, is the deeper one and represents the global minimum of the conceptual function, while other valleys are local minima.

Figure 6 illustrates two types of arrows: solid red line and dashed blue line both with an arrowhead. The former is related to the gradient direction, i.e., the particle is using the gradient information rather than following the global tendency. The latter indicates the particle is following the global best particle. Also, a red letter **x** indicates the global best position, which pushes the particles to its position during the search process.

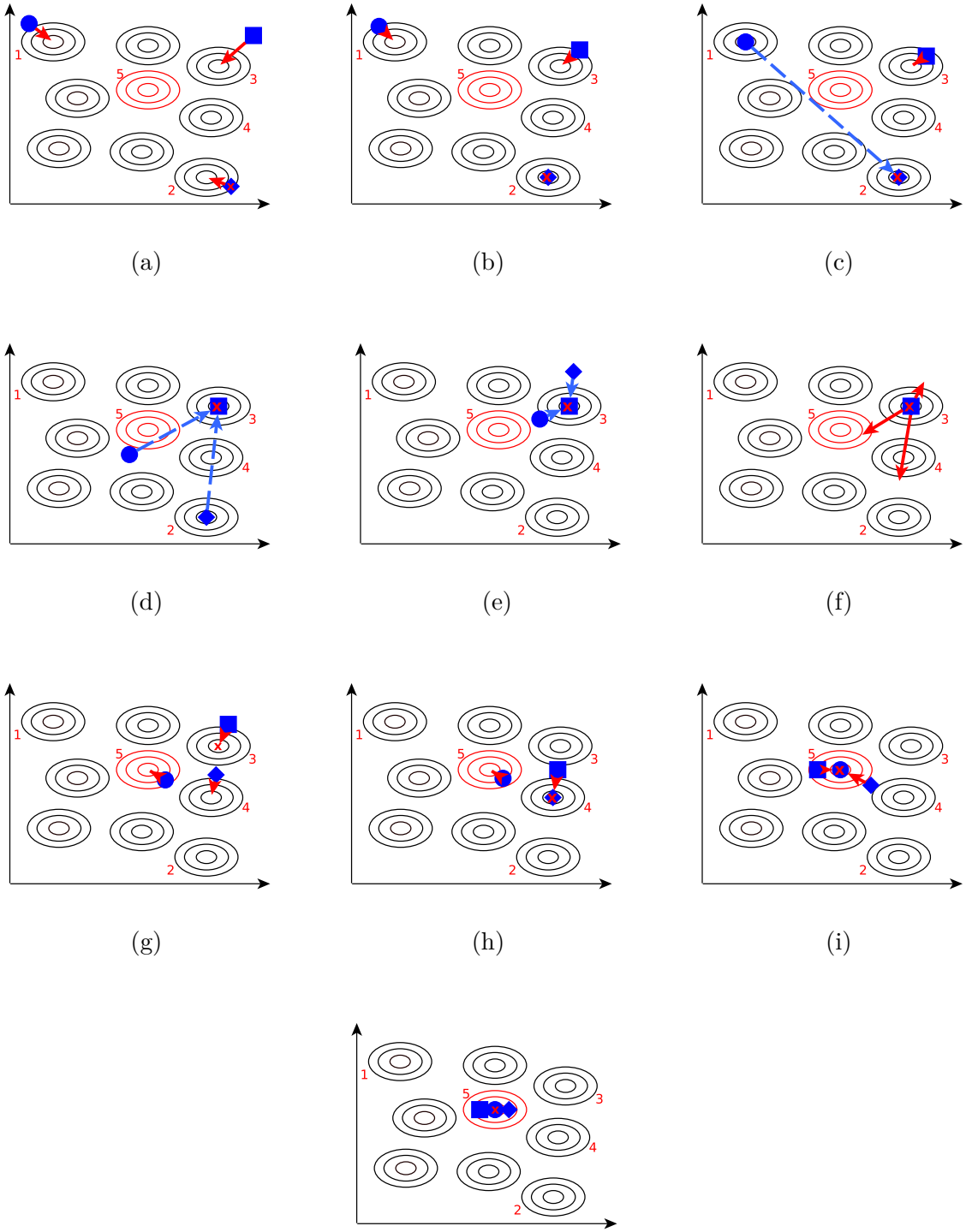
Figure 6a shows the three particles using their gradient direction to investigate their neighborhoods. All particles are in the attraction phase. As the valleys 1, 2, and 3 are nearest the particles, Figure 6b highlights the first local minimum found by the diamond particle, whereas the circle and square particles are on their ways to other local minima. Figure 6c illustrates that the circle particle will start to follow the current best position, whilst the square particle is approaching the best position in its neighborhood.

Figure 6d characterizes another local minimum found by the square particle. The circle particle was about to follow the diamond particle, while a deeper valley was found by the square particle. Thus, both circle and diamond particles are now on their ways to find the position where current best particle is. Note by the Figure 6e, the circle particle is very close to the current best particle, and the diamond particle has flown over the best current position. Finally, Figure 6f presents the repulsive scheme where the particles are about to repel each other.

After the repulsion design takes place, the particles active their gradient information and attraction phase again. This behavior is presented in the Figure 6g. Mainly because of the repulsive process, the square and diamond particles fell near the valleys 4 and 5, while the square particle fell at the same valley 3. One can note in Figure 6h that the diamond particle was the first one to find another valley, thus this particle turns into the current global best solution. It is also easy to imagine that the square particle found later the same local minimum (valley 3) and after decided to follow the global tendency. However, while the square particle was moving in the direction of the current best position, the circle particle finally found the global minimum of the conceptual function. Therefore, both square and diamond particles are moving to the global best position, this is depicted by Figure 6i.

Lastly, Figure 6j shows a clustering of particles. After this stage, the particles would repel each other again likely the one presented in the Figure 6f. However, the global memory is saved at the valley 5. Thus, even if another valley is found after the repulsive phase, the particle will surely come back to the valley 5, because no other valley is deeper than valley 5 in that conceptual function. Of course, in a real-world optimization problem, the common is to not know where the global minimum is, thus the repulsive design is a very important feature incorporated in the SAPSO algorithm.

Figure 6 – A short self-contained example of SAPSO algorithm with three particles. The numbers near each representative valley are indications of the valley depth. The higher the number associated to the valley, the deeper the valley is.



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

(j)

Source: produced by the author.

3.1.2 The velocity update equation

As previously mentioned, each particle of the swarm can decide between follow the negative direction of the gradient or follow the global tendency of the swarm, i.e., fly in the direction of the global best particle. One binary variable encapsulates the decision of the ith particle, such that $I_i \in \{0, 1\}$. To verify this binary variable at each iteration and for each particle would introduce a small but inconvenient procedure in the algorithm. A clever way to embody such idea into the velocity update equation without checking binary variables is as follows:

$$\vec{\mathbf{v}}_i^{t+1} = w^t \vec{\mathbf{v}}_i^t + dir \times \Big[\underbrace{I_i^t c_1 \phi_1 \big(\vec{\mathbf{g}}^t - \vec{\mathbf{x}}_i^t\big)}_{\text{social component}} + \underbrace{\big(I_i^t - 1\big) c_2 \phi_2 \nabla f(\vec{\mathbf{x}}_i^t)}_{\text{gradient component}} \Big]. \tag{3.1}$$

One can note that when $I_i = 0$, the so called social component will be zero and the gradient component is applied in the ith particle along with the momentum, i.e., a portion of the velocity from the last iteration. The gradient component places the particle in the direction of the steepest descent from the objective function, since the algorithm is minimizing the objective function. When $I_i = 1$, no gradient component is applied, but instead, the social component takes place.

As shown extensively in (KENNEDY, 1997a), the social-only version of the PSO algorithm (no cognitive term is used to update velocity) performs better than the cognitive-only model (without the social term) and full model (both cognitive and social terms are applied). As can be seen in the Equation 3.1, the social-only model is preferred in the stochastic search phase, since the cognitive-only or full model would not have much significance in this context. The cognitive-only would duplicate the information about local neighborhood area of the search space with previous best position and gradient information. This approach would lead to a premature convergence sooner. The full model would incorporate more information than necessary, once four components is enough to the motion of a single particle.

Since the velocities are updated stochastically, the trajectory of a particle may cross the boundary of the search space and it may fall into a state of "velocity explosion" (CLERC; KENNEDY, 2002). To overcome this problem, SAPSO algorithm uses a inertia weight, a velocity limitation and a gradient clamping. The same approach is used to clamp velocity and gradient vectors, which is the one described earlier as a velocity threshold in the Equation 2.5. Herein, v_{max} is defined according to

$$v_{max} = \frac{|x_{max} - x_{min}|}{2} \tag{3.2}$$

where x_{max} and x_{min} correspond to the maximum and minimum values of the variable. This velocity clamping poses a limit to be maximum velocity of each particle, preventing it from moving too far beyond the limits of the search space.

3.1.3 The algorithmic steps

In the beginning of the algorithm, each particle is uniformly spread through the multidimensional search space. The swarm starts with $I = [0, \ldots, 0]$ and $dir = 1$, which means that each particle must investigate its own area through the negative direction of gradient and the swarm is in the attraction phase, respectively. The nearest local minimum (or global minimum) will be reached individually by the particles, as explained in Section 2.6. It is worth noting that each particle will converge to the local minimum closest to its initial position, and only one of those local minima will be the global best position $\vec{\mathbf{g}}^t$ among the particles at some iteration t.

The ith particle takes the decision to switch from local investigation to follow the global tendency when $I_i = 0$ turns to $I_i = 1$. A trigger to determine the right time this decision has to be applied is to analyse the first order derivative of the function at $\nabla f(\vec{\mathbf{x}}_i^t) = 0$. However, the gradient is calculated through an approximation of partial derivatives by a forward-differencing (see Equation 2.17), thus another approach is preferred instead of looking for a critical point.

A straightforward process to detect whether a particle is trapped in a local minimum is to evaluate if its fitness stagnates during sequential iterations. The ith particle has a counter C_i which detects quite small changes of its fitness value. It seems acceptable to evaluate this condition as follows:

$$|f(\vec{\mathbf{x}}_i^t) - f(\vec{\mathbf{x}}_i^{t-1})| \leq \epsilon, \tag{3.3}$$

where ϵ is a small threshold (e.g., 10^{-2}). Whenever this condition is satisfied, the counter is incremented, otherwise the counter resets to zero. When a predetermined $cMax$ parameter is reached, the particle is at the bottom of a local minimum and it is time to follow the tendency of the global best position, thus $I_i = 1$ and $cMax = 0$ again.

The ith particle takes the decision to switch back from global investigation to follow gradient direction again when $I_i = 1$ turns to $I_i = 0$. This decision only happens if the particle is near enough (ne) to the global best particle (e.g., 10^{-5}). Algorithm 5 describes the aforementioned approach about the internal decision made by each particle. The DIST$(\cdot, \cdot)$ function calculates the Euclidean distance between two vectors.

Another important note is if the local minimum found by the ith particle is the current best position found so far. After $I_i = 1$ turns to $I_i = 0$, the ith particle will continue at the same area and the candidate solution will still be improved by the gradient information again.

To summarize the steps of the SAPSO algorithm, Figure 7 shows the flowchart with all the important phases of the proposal. It is worth pointing out that the step "update global best position G" is able to update the global best position of the swarm by the ith

Algorithm 5 Particles' internal decision algorithm.

```
1: function AutonomousDecision(x⃗, I, C, g⃗)
2:     for i = 1 to n do
3:         if I_i == 0 then
4:             if |f(x⃗_i^t) − f(x⃗_i^{t−1})| ≤ ϵ then
5:                 C_i = C_i + 1
6:                 if C_i == cMax then
7:                     I_i ← 1
8:                     C_i ← 0
9:                 end if
10:            else
11:                C_i ← 0
12:            end if
13:        else
14:            if dist(x⃗_i, g⃗) ≤ ne then
15:                I_i ← 0
16:            end if
17:        end if
18:    end for
19:    return I, C
20: end function
```

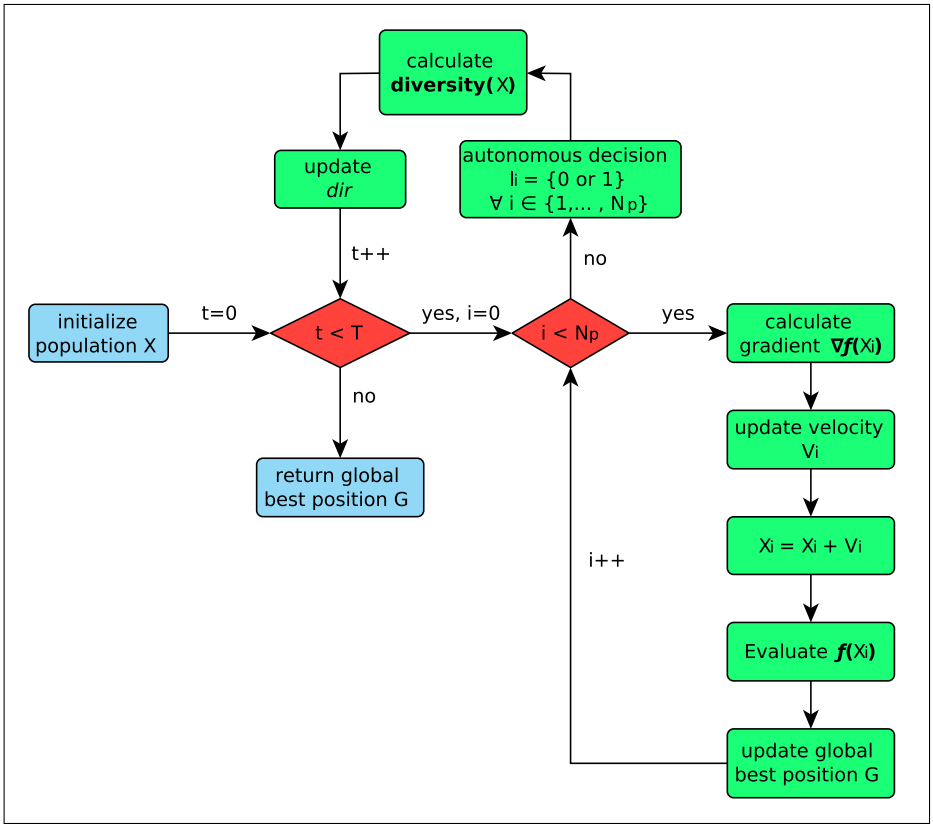
particle. In this case, the next $(i+1)$th particle will immediately use the new global best position in its calculus of velocity and that situation may happen at the same iteration. This strategy brings a faster communication among the particles, as there is no need to wait for a full iteration to use the new global best position. The GPSO algorithm uses that same strategy, whereas ARPSO and DGHPSOGS algorithms wait for a full iteration to update the global best position among the particles. Also, as mentioned before, both autonomous decision and phase of attraction or repulsion are chosen after an iteration is finished in the steps "autonomous decision" and "update *dir*", respectively.

3.1.4 Exploration versus exploitation: a contradictory trade-off

Any metaheuristic must embed exploration and exploitation features. They are both equally important and should be explored in a clever manner during the algorithm execution time. However, in (VESTERSTRØM; RIGET, 2002), the authors raise the contradiction of maintaining high diversity and obtaining fast convergence as both goals of any metaheuristic. In addition, the problem with premature convergence often persists in multimodal optimization, which results in great performance loss and suboptimal solutions. The clustering of particles in classical PSOs seems to be formed by the fast information flow among the particles, resulting in serious issues, such as difficulties of escaping local optima, low diversity of the population, and fitness stagnation, i.e., the clustering might lead to a suboptimal solution as an overall result.

As the optimization tasks are NP-hard problem, the practitioners should check

Figure 7 – Flowchart of the SAPSO algorithm.

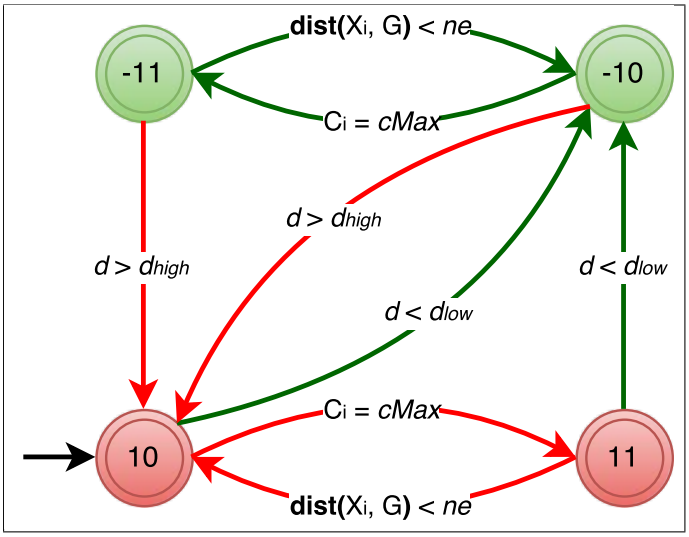


Source: produced by the author.

the whole continuous search space to ensure that a result is not suboptimal, and this task would lead to a search with prohibitive time. It seems to be unanimous and to make perfectly sense to improve the optimization algorithms to avoid suboptimal solutions more frequently. It is also worth noting that the key to address multimodal optimization problems and the aforementioned issues seems to be an approach that efficiently alternates between phases of global exploration and phases of local exploitation during the execution time, instead of just implement a transition from exploration to exploitation as is common done throughout the literature, specially with those time-dependent approaches, such as linear decreasing of the inertia weight (SHI; EBERHART, 1998a; RATNAWEERA; HALGAMUGE; WATSON, 2004; ARUMUGAM; RAO, 2006). This issue is one of the fundamental principles of the SAPSO algorithm proposed in this thesis. Once the PSO saves the current best position (and the practitioner does not know whether it is the global minimum) after a convergence has happened, a divergence can be applied in the swarm without any loss of information. It sounds like a new chance to find another prominent area of the search space is given to the particles on the same course of the algorithm execution.

The background idea of the SAPSO algorithm is to properly investigate the search space and to avoid premature convergence. The algorithm also escapes from local minima,

Figure 8 – The SAPSO algorithm during runtime. The behavior of the method is expressed by autonomous decisions of each particle and by social decisions of the swarm.



Source: produced by the author.

which is controlled by the diversity measure through a repulsive design. In Figure 8, a particle's view of the algorithm along with the social decision of the swarm is depicted. As can be seen, four states (presented as circles) are depicted, representing the states that each particle of the SAPSO algorithm can assume in the search process. Each state controls the way a particle will move through the search space. The four states are represented by two binary variables: dir and I_i. After each iteration, the particles can change their internal state through the transition analysis (presented as arrows). Note that, for simplicity purpose, the transitions that connect a state to itself are omitted. The analysis of the diversity value can change the state of the particles as well.

The behavior of the entire population when the method starts running is to locally investigate the search space, i.e., each particle is attentive to its own gradient information. This is expressed by the state $\{10\}$ (the left arrow without source circle represents the initial state), which are $dir = 1$ and $I_i = 0$, respectively. The two important choices are taken at each iteration of the algorithm, one choice is taken by the particle and the other one is taken by the swarm. After a period of time, the global best position is already set and each particle individually decides the moment to change its perspective of searching towards that position. The change is controlled by a counter C_i, which is incremented every time a sequential fitness evaluation results in almost the same fitness value as the previous iteration. By the time the maximum number of local evaluations ($cMax$) with almost the same fitness value is reached, the transition $C_i = cMax$ is applied, the counter C_i is reset to zero. The particle will change its behavior to the state $\{11\}$, thus it remains $dir = 1$ and turns $I_i = 1$.

When $I_i = 1$, the gradient information is disregarded and the particle follows the

tendency of the current global best position found so far. By the time the position of the particle is near enough (ne) of the global best position, its behavior changes again to the state {10}. This is the same previous behavior presented in the Algorithm 5. It is important to note that both states ({10} and {11}) become quickly interchangeable among the particles, as they will be flying at the same valley of the global best position, even tuning and improving $\vec{\mathbf{g}}$ a little, particle by particle, iteration by iteration. Of course, while a single particle is flying in the direction of the global best position, another prominent valley might be discovered. In this case, the swarm will move to that new position over the next iterations.

As the particles approach each other through the attractive design ($dir = 1$), the diversity measure becomes lower. This happens when each particle pass through the state {11} at least once. The convergence trend to one of the local minima (or a global minimum) becomes stronger at each subsequent iteration. This behavior of convergence is desirable for fine-tuning purpose, however it may trap the entire population in a local minimum. Thus, to avoid the local minimum as a result, the metaheuristic controls the diversity of the entire population. When the diversity value becomes less than a predefined value (d_{low}), the whole population changes to the state {-10}. Now $dir = -1$, which indicates a repulsive design of the particles.

Instead of following the global tendency, the particles follow their own negative gradient directions. Note that $I_{1,\ldots,n} = 0$ only when the transition from the state {11} (or {10}) to the state {-10} is applied. Also note that, although the swarm is in a repulsive phase, a single ith particle can alternate between the states {-10} and {-11} at any time according to C_i value or the distance between the ith particle and the global best position. The repulsive behavior will increase the diversity of the population and spread the particles throughout the search space again. In this case, the desirable diversity is controlled by a predefined d_{high} value, which the entire population will change its behavior to the state {10} again. In other words, each particle is far apart from each other and will investigate its local area of the search space over again through the gradient information. Additionally, $I_{1,\ldots,n} = 0$ only when the transition from the state {-10} (or {-11}) to the state {10} is applied.

The state {-10} (or {-11}) is peculiar for two reasons: it guarantees the avoidance of local minima and it also contributes to the exploration of the search space. Observe that even if the local minimum previously investigated is actually the global minimum, there is no problem in spreading the particles through the search space again, as the global best position found so far is saved. Thus, if no interesting area is found when repulsive design is applied, the particles will mostly return to the global best position ever found.

3.2 Rotation and information exchange

In the context of PSO applications, the general focus of the works is about solving specific optimization problems, which is generally the central contribution of the practitioners' works. The precise information of how particles move is often scarce. Due to the lack of this information, it is not self explanatory how random numbers are drawn, whether they are scalars or vectors. Both approaches guide the particle movement in a different manner. Herein, two types of equations are investigated: rotation invariance (scalar random numbers already showed in Section 2.8.1) and rotation variance (vector of random numbers previously discussed in Section 2.8.2).

The movement of the ith particle is described by the velocity vector $\vec{\mathbf{v}}_i$. The rotationally variant PSO version is redefined here according to the following velocity update equation

$$\vec{\mathbf{v}}_i^{t+1} = w^t \vec{\mathbf{v}}_i^t + c_1 \dot{\vec{\boldsymbol{\phi}}}_i^t \odot \left(\vec{\mathbf{p}}_i^t - \vec{\mathbf{x}}_i^t\right) + c_2 \ddot{\vec{\boldsymbol{\phi}}}_i^t \odot \left(\vec{\mathbf{g}}^t - \vec{\mathbf{x}}_i^t\right), \tag{3.4}$$

where $\dot{\vec{\boldsymbol{\phi}}}_i^t$ and $\ddot{\vec{\boldsymbol{\phi}}}_i^t$ are two uniformly distributed random vectors between $[0, 1]$ at iteration t (this equation is equivalent to the one with diagonal random matrices $\boldsymbol{\Phi}_{l=1,2}$ defined in Section 2.8.2), and w^t is the inertia weight described in (SHI; EBERHART, 1998a). This last parameter balances the trade-off between global and local search. The time-varying inertia weight can be a positive constant, linear or nonlinear function of time. The operator $\odot$ stands for the element-wise multiplication, also known as Hadamard product (HORN; JOHNSON, 2012). In this scenario, the element-wise multiplication embeds a wider range motion for each particle. During the search process, while the swarm converges to promising regions, the particles are encouraged to explore areas slightly deviated from the direction of their local and global memories.

As a reminder, the velocity update equation related to the invariant version of PSO is redefined as follows

$$\vec{\mathbf{v}}_i^{t+1} = w^t \vec{\mathbf{v}}_i^t + c_1 \dot{\phi}_i^t \left(\vec{\mathbf{p}}_i^t - \vec{\mathbf{x}}_i^t\right) + c_2 \ddot{\phi}_i^t \left(\vec{\mathbf{g}}^t - \vec{\mathbf{x}}_i^t\right), \tag{3.5}$$

where $\dot{\phi}_1^t$ and $\ddot{\phi}_2^t$ are two uniformly distributed random numbers between $[0, 1]$ at iteration t. Note that, this PSO version limits the movement of a particle to fly over combinations of $\vec{\mathbf{p}}$ and $\vec{\mathbf{g}}$. To make this distinction clear, the next sections reports a graphical demonstration of the search distribution by considering both approaches.

3.2.1 Graphical demonstration of the instantaneous search domain

To illustrate different aspects of each PSO version, Figure 9[1] depicts specific combinations of the searching vectors $(\vec{\mathbf{p}}_i - \vec{\mathbf{x}}_i)$ and $(\vec{\mathbf{g}} - \vec{\mathbf{x}}_i)$ related to a particle $\vec{\mathbf{x}}_i$. Herein,

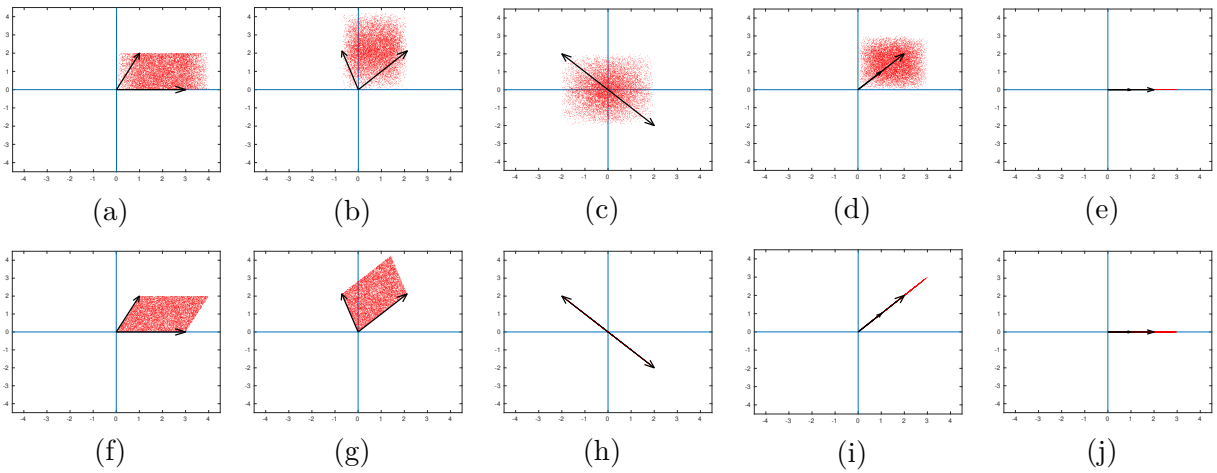
[1] The velocity component where inertia weight appears is present in both versions of PSO. Therefore, the figures and discussion in this section disregard this component.

search distribution is relative to the Distribution of all Next Possible Position (DNPP) in (CLERC, 2012b), which is represented by sample points. Figures 9a to 9e show rotationally variant examples and Figures 9f to 9j illustrate rotational invariant examples generated using Monte Carlo simulations with 10^4 samples of $\vec{\dot{\boldsymbol{\phi}}}_i \odot (\vec{\mathbf{p}}_i - \vec{\mathbf{x}}_i) + \vec{\ddot{\boldsymbol{\phi}}}_i \odot (\vec{\mathbf{g}} - \vec{\mathbf{x}}_i)$ and $\dot{\phi}_i(\vec{\mathbf{p}}_i - \vec{\mathbf{x}}_i) + \ddot{\phi}_i(\vec{\mathbf{g}} - \vec{\mathbf{x}}_i)$, respectively.

In Figures 9a and 9f, the vectors are given by $(\vec{\mathbf{p}}_i - \vec{\mathbf{x}}_i) = [1, 2]$ and $(\vec{\mathbf{g}} - \vec{\mathbf{x}}_i) = [3, 0]$. In order to test a different perspective, Figures 9b and 9g show the search distribution of vectors after rotating in 45° counterclockwise given by $(\vec{\mathbf{p}}_i - \vec{\mathbf{x}}_i) = [-\frac{\sqrt{2}}{2}, \frac{3\sqrt{2}}{2}]$ and $(\vec{\mathbf{g}} - \vec{\mathbf{x}}_i) = [\frac{3\sqrt{2}}{2}, \frac{3\sqrt{2}}{2}]$. The following examples are specific situations to evaluate the search distribution when: cognitive and social components have the same lengths with opposite directions (Figures 9c and 9h), respectively, $(\vec{\mathbf{p}}_i - \vec{\mathbf{x}}_i) = [-2, 2]$ and $(\vec{\mathbf{g}} - \vec{\mathbf{x}}_i) = [2, -2]$; cognitive and social components with the same directions (Figures 9d and 9i), respectively, $(\vec{\mathbf{p}}_i - \vec{\mathbf{x}}_i) = [1, 1]$ and $(\vec{\mathbf{g}} - \vec{\mathbf{x}}_i) = [2, 2]$; and finally cognitive and social components with the same directions parallel to x-axis (Figures 9e and 9j) with $(\vec{\mathbf{p}}_i - \vec{\mathbf{x}}_i) = [1, 0]$ and $(\vec{\mathbf{g}} - \vec{\mathbf{x}}_i) = [2, 0]$, respectively.

Generally, one can see a clear difference between the search distributions of rotational variant and rotational invariant versions of PSO. In Figure 9a, the probability distribution follows a trapezoidal shape due to different random numbers mutiplied by each component of each searching vector, whereas in Figure 9f the probability distribution is uniform, as the same random number is multiplied by both components of each searching vector, thus changing its magnitude only. After rotating 45° counterclockwise, one can note this rotation severely modifies the search distribution of samples in Figure 9b, whereas the rotation just rotates the uniform distribution of samples in Figure 9g. As shown in

Figure 9 – Search distribution of all possible next positions given by searching vectors $(\vec{\mathbf{p}} - \vec{\mathbf{x}})$ and $(\vec{\mathbf{g}} - \vec{\mathbf{x}})$. The particle sample is at the origin $\vec{\mathbf{x}} = [0, 0]$ and a total of 10^4 samples make part of the search distribution.



Source: produced by the author.

Figure 9c, both searching vectors have the same length with opposite directions and the probability distribution follows a triangular shape, which guarantees directional diversity. On the other hand, there is no directional diversity in Figure 9h, as the search distribution collapses along the searching vectors, which in turns is not the desired behavior, specially in the beginning of the algorithm where the exploration should be more prioritized than exploitation. Similar case is shown in Figures 9d and 9i. The directional diversity in the rotation invariance vanishes when both searching vectors are parallel to each other. After rotating the parallel searching vectors 45° clockwise (Figures 9e and 9j), both search distributions collapse on one line. In this scenario, no distinction is observed in relation to the search distribution, but it is also worth to point out that the rotationally variant version of velocity update equation (Figure 9e) is prone to a bias phenomenon (SPEARS; GREEN; SPEARS, 2010) where the particles have some predilection in making movements parallel to one of the coordinate axes. In this scenario, the particles would be trapped in making movements parallel to the x-axis until a new prominent area of the search is discovered or a stop criteria is matched.

3.2.2 Empirical average angle analysis

According to (KENNEDY, 2007) apud (SPEARS; GREEN; SPEARS, 2010), James Kennedy has indicated that the variant version should be preferred, as it is considered to be more exploratory. To ratify the aforesaid intuition, two experimental test were performed. The goal is to perform both versions of PSO algorithm in test functions and evaluate the average angle between the real ($\vec{\mathbf{r}}$) and the estimated ($\vec{\mathbf{e}}$) vectors, respectively, considering and not considering random numbers in the process of updating the velocity equation. Figure 10 shows the angle θ under consideration. One may expect that the average angle of variant version is higher than the invariant one.

The angle is achieved according to

$$\theta = \arccos\left(\frac{\vec{\mathbf{e}}_i^t \bullet \vec{\mathbf{r}}_i^t}{||\vec{\mathbf{e}}_i^t||\ ||\vec{\mathbf{r}}_i^t||}\right), \tag{3.6}$$

where

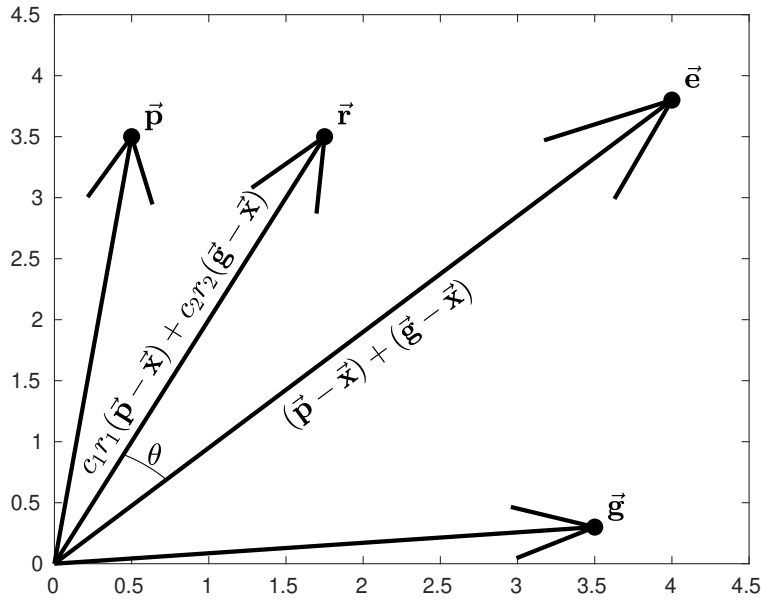
$$\vec{\mathbf{e}}_i^t = \left(\vec{\mathbf{p}}_i^t - \vec{\mathbf{x}}_i^t\right) + \left(\vec{\mathbf{g}}^t - \vec{\mathbf{x}}_i^t\right) \tag{3.7}$$

and

$$\vec{\mathbf{r}}_i^t = \begin{cases} c_1\vec{\dot{\boldsymbol{\phi}}}_i^t \odot \left(\vec{\mathbf{p}}_i^t - \vec{\mathbf{x}}_i^t\right) + c_2\vec{\ddot{\boldsymbol{\phi}}}_i^t \odot \left(\vec{\mathbf{g}}^t - \vec{\mathbf{x}}_i^t\right) & \text{for variant version,} \\ c_1\dot{\phi}_i^t\left(\vec{\mathbf{p}}_i^t - \vec{\mathbf{x}}_i^t\right) + c_2\ddot{\phi}_i^t\left(\vec{\mathbf{g}}^t - \vec{\mathbf{x}}_i^t\right) & \text{for invariant version.} \end{cases} \tag{3.8}$$

The experimental tests were conducted over four well-known test functions: Ackley ($\mathbb{S} = \{x \mid -50 \leq x \leq 50\}$), Griewank ($\mathbb{S} = \{x \mid -100 \leq x \leq 100\}$), Rosenbrock ($\mathbb{S} = \{x \mid -5.12 \leq x \leq 5.12\}$), and Rastrigin ($\mathbb{S} = \{x \mid -5.12 \leq x \leq 5.12\}$). Two experiments are conducted with different configurations: 1) inertia weight is a linear

Figure 10 – The angle θ between the estimated position $\vec{\mathbf{e}}$ reached by a velocity update equation without influence of random numbers, and the real position $\vec{\mathbf{r}}$ formed by a velocity update equation with random numbers. The random structure $\vec{\mathbf{r}}_i$ can be $\vec{\boldsymbol{\phi}}_i$ or ϕ_i.



Source: produced by the author.

decreasing function of time, varying from 0.9 down to 0.4. Moreover, cognitive and social coefficients are fixed in 2; and 2) inertia weight is 0.7298, while cognitive and social parameters are set as 1.4962, which is similar to the ones described in (CLERC, 2012b). The PSO settings are described in Table 2. Different numbers of dimensions are configured to evaluate whether the average angle is affected by the high number of variables. Furthermore, the high number of iterations is intentional and serves to find out the effect of small movements of particles after the convergence of the swarm.

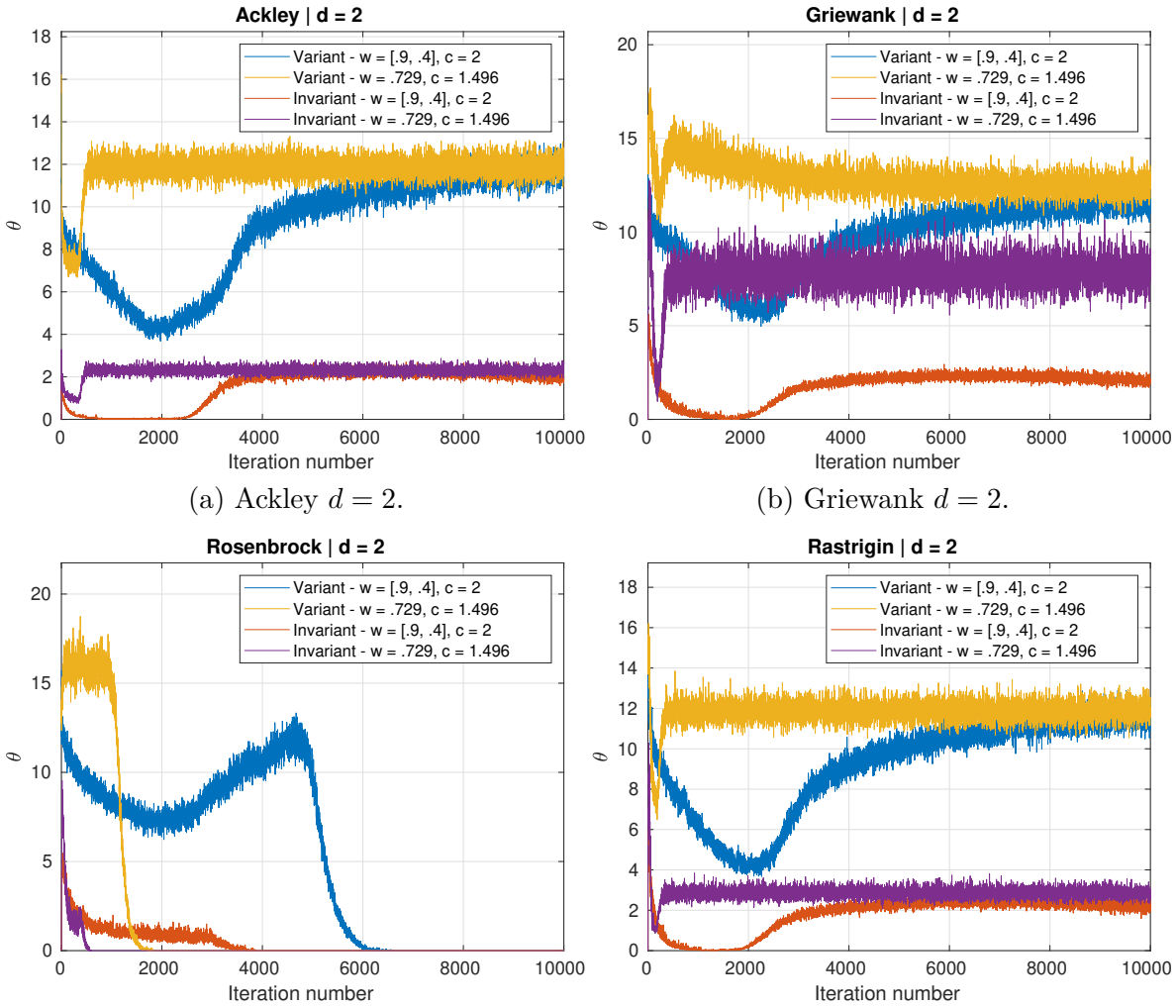
Table 2 – PSO settings for variant and invariant versions. Different coefficients are used in each configuration.

Variable	**Conf. 1**	**Conf. 2**
Number of particles (n)	20	20
Number of dimensions (d)	[2, 10, 30, 50]	[2, 10, 30, 50]
Cognitive coefficient (c_1)	2	1.4962
Social coefficient (c_2)	2	1.4962
Inertia weight (w)	[.9 $\rightarrow$ .4]	.7298
Number of iterations (T)	10000	10000
Number of executions (E)	50	50

Source: produced by the author.

Figure 11 presents the results based on the average angle of the swarm in each

Figure 11 – Average θ of the swarm according to the movement of particles along the iterations. Figures for high dimensions are depicted in Appendix A.



(a) Ackley $d = 2$. (b) Griewank $d = 2$.

(c) Rosenbrock $d = 2$. (d) Rastrigin $d = 2$.

Source: produced by the author.

iteration for both versions. The author summarizes the analysis in the two dimensional space, since for high dimensions the behavior of average angle is rather similar (see Appendix A to visualize figures containing the average angles in higher dimensions for each function). In a general view, one can see that the average angle of movement along the iterations in the variant version is higher than the invariant one, which means the search mechanism of this version is more explorative.

Configuration 1 with values of $w = 0.9 \rightarrow 0.4$ and $c_1 = c_2 = 2$ demonstrates a decay and a growth in the value of average angle during the search process in both versions of the algorithm, which is probably the process of swarm convergence, i.e., the exploration phase decreases the average angle while the particles strongly follow the global tendency and they are less influenced by their local memories. In the beginning of the algorithm, it is expected that the swarm starts a process of convergence toward the global best position.

If a particle $\vec{\mathbf{x}}_i^t$ updates its local memory $\vec{\mathbf{p}}_i^t$ at iteration t, then at the iteration $t+1$, the cognitive component $(\vec{\mathbf{p}}_i^{t+1} - \vec{\mathbf{x}}_i^{t+1})$ becomes a zero vector $\vec{\mathbf{0}}$. In this situation, the social component plays a total role in the directional diversity of the next particle's movement. Moreover, this scenario reduces drastically the search distribution and it is probably the reason why there is a decline in the average angle values at early iterations. Nevertheless, the exploitation phase around the global best position found so far increases the average angle values, since both cognitive and social components are now actuating on the particles' movement. The type of micro-searching process, i.e., small movement of particles after a prominent area is discovered by the best particle, is a phenomenon observed in all test functions with different number of dimensions.

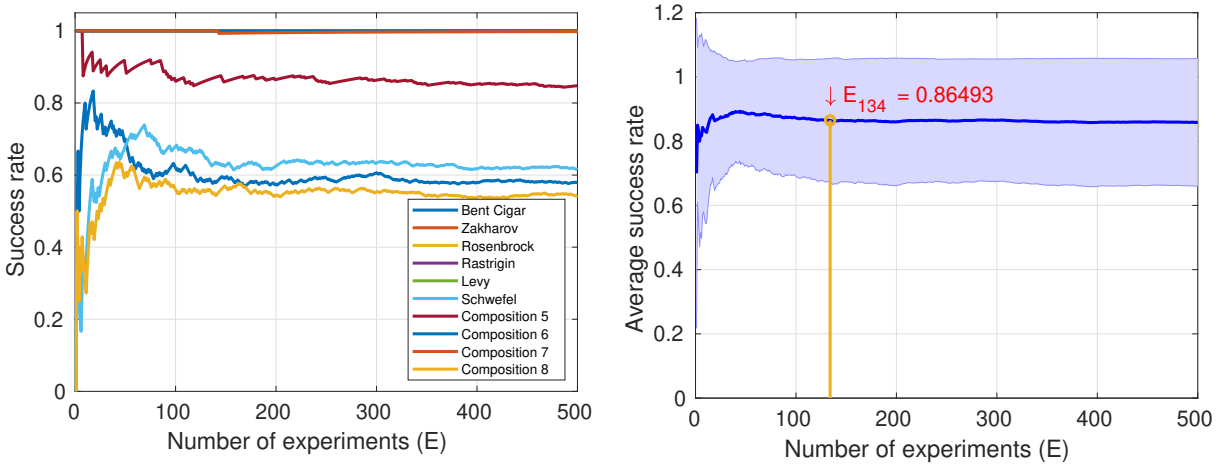
Configuration 2 fixed values of $w = 0.7298, c_1 = 1.4962$, and $c_2 = 1.4962$, the average angle is well-behaved during the whole search process, independently of the PSO version. The only exception happens in the Rosenbrock function with $d = 2$, where the values of angles reduce drastically after a few iterations. For a low dimension, this test function is non-convex and unimodal with global minimum value displaced from the center of the search space. This contributes to the search direction being trapped in only one direction, which is pointing to where the global best position is. As a consequence, one can observe the average angle approaching zero value, which means the real vector $\vec{\mathbf{r}}$ is parallel to the estimated vector $\vec{\mathbf{e}}$. Note that for high dimensions (Appendix A), this behavior is less noticeable and the average angle values are similar to the ones observed in other test functions.

There is no evidence of relation between curves of average angle and the number of iterations or different number of dimensions. Most of the time, the angles of variant version prevails between 20° and 25°, whereas the invariant maintains below 10°. These findings are consistent with the argument stated by Kennedy and it might be an indicative that the rotationally variant PSO version explores a broader range of the search space, while converging to promising areas. On the other hand, the rotationally invariant PSO version is prone to lean towards the estimated vector $\vec{\mathbf{e}}$ most of the time, prevailing the exploitation due the covering of a narrow region of the search space.

3.2.3 A method to define the number of executions

In this section, a general method to determine the minimum number of executions (E) is presented. Since the expected value (μ) of a measured performance (e.g., success rate and minimum value of a function) and the variance (σ^2) can not be previously defined before algorithm's run (see Rule 2 defined by Clerc and discussed in Section 2.10), another approach must be investigated to select a suitable value for E. The success rate is a performance criterion to determine the achievement of an algorithm, i.e., whether the goal of the optimization problem is found or not. A simple manner to evaluate the success rate

Figure 12 – Number of experiments based on CEC 2017 optimization problems using a rotationally variant PSO version with late information exchange among particles.



(a) Success rate in each function.

(b) Average number of experiments.

Source: produced by the author.

is formulated by

$$s_{ij} = \begin{cases} 1 & \text{if } f_i(\vec{\mathbf{x}}) - f_i(\vec{\mathbf{x}}_{opt}) \leq \xi \\ 0 & \text{else} \end{cases} \tag{3.9}$$

and

$$sr_{ij} = \frac{1}{j} \sum_{z=1}^{j} s_{iz} \mid \forall j \in \{1, 2, \ldots, E^{\star}\}, \tag{3.10}$$

where s_{ij} is the achievement of the algorithm regarding an optimization function $f_i(\cdot)$ with global minimum at $\vec{\mathbf{x}}_{opt}$, $sr_{ij} \in [0, 1]$ is the actual success rate specified by the number of executions j, ξ is a predefined error threshold empirically defined as $\xi = 10^{-4}$, and $E^{\star}$ stands for the maximum number of executions. To find the appropriate number of executions, the applicability of the above methodology refers to the variant version of PSO with late exchange of information among particles.

Figure 12a highlights the success rate of the algorithm applied in the set of test functions from CEC 2017 benchmark suite described later in Section 4.2.1. One can clearly infer that a small number of executions (e.g., $E < 50$) is not enough to produce reliable results, as the curves are highly unstable. For instance, with $E < 25$, the success rate in Bent Cigar function is above 80%. However, with $E > 100$, the success rate is stabilizing below 60%. Small values of E might indicate results with high success rate by chance, whereas high values considerably decrease this random effect.

A general behavior must be considered if more than one optimization test function

is used to evaluate the algorithms. Thus, the average value of each curve is measure by

$$\overline{sr}_i = \frac{1}{E^\star} \sum_{j=1}^{E^\star} sr_{ij}, \tag{3.11}$$

where $E^\star = 500$ in Figure 12, $\overline{sr}_i$ is the average number of success rate in the test function $i \mid \forall\, i \in \{1, 2, \ldots, m\}$, and m is the number of functions.

The minimum number of executions for each test function according to the maximum deviation ϵ is discovered as follows

$$e_i \overset{j}{\leftarrow} |sr_{ij} - \overline{sr}_i| \leq \epsilon, \forall\, j \in \{1, 2, \ldots, E^\star\}, \tag{3.12}$$

where $e_i = j$ when the success rate curves stabilize. Note that, other $j' > j$ may satisfy the above inequality, however the algorithm stops after j is discovered. Several values for ϵ were tested during experiments, the author suggests $\epsilon = 10^{-4}$ as an initial guess for general purpose. Finally, the average of all number of executions is defined by

$$E = \frac{1}{m} \sum_{i=1}^{m} e_i. \tag{3.13}$$

It is worth to point out that the exact number of executions embeds the average behavior of all test functions, i.e., it represents the number of executions where the results for all test functions may be considered stable and reliable.

Algorithm 6 details the proposed method to define the average number of executions. The inside loop (lines 4–12) performs Equations 3.9 and 3.10, respectively, the if/else statement (lines 5–9) saves the achievement of ith function in jth execution, and line 11 performs the current jth success rate. Line 13 simply performs the average of the values in sr_i. The second inside loop (lines 14–19) carries out the number of executions required to stabilize the success rate curve of the ith function. The jth position selected, which is the required number of executions, corresponds to the success rate value closest to the average success rate value. The loop breaks as soon as this number is discovered (line 17). Finally, the result of the algorithm is the average number of required executions for each optimization problems (line 20).

The aforementioned method applied to the set of test functions is presented in the Figure 12b. The average success rate shows the stabilized curve whose the minimum number of executions found by the Equation 3.13 is $E = 134$. The result of this experiment will be used as a reliable number of executions to analyse the comparison among PSO versions later in Section 4.2.

3.3 ISAPSO: Invariant SAPSO

This section proposes adjustments for SAPSO algorithm principally in terms of finding better solutions (e.g., low error rates) and reduction of computational time. In

Algorithm 6 Algorithm to find the average number of executions.

```
1: function nExec(ϵ, ξ, m, E*)
2:     for i = 1 to m do
3:         sum ← 0
4:         for j = 1 to E* do
5:             if |f_i(x⃗) − f_i(x⃗_opt)| ≤ ξ then
6:                 s_ij ← 1
7:             else
8:                 s_ij ← 0
9:             end if
10:            sum ← sum + s_ij
11:            sr_ij ← sum/j
12:        end for
13:        sr̄_i ← AVG(sr_i)
14:        for j = 1 to E* do
15:            if |sr_ij − sr̄_i| ≤ ϵ then
16:                e_i ← j
17:                break
18:            end if
19:        end for
20:    end for
21:    E ← AVG(e)
22:    return E
23: end function
```

addition, a mathematical proof is given to show that ISAPSO is *strictly* rotationally invariant when the swarm is in the attraction phase, while the algorithm is rotationally invariant in a *stochastic sense* in the repulsion phase. The motivations to develop this new version comes from the theoretical and empirical studies provided by Sections 2.8, 2.9, 3.1, and 3.2 about rotational variance and invariance, separable and non-separable functions, directional diversity, and biases toward the coordinate system. To work with rotation invariance property is rather preferable than facing a bias phenomenon with rotation variance property. These topics has inspired a development of a new SAPSO version which is advisable to be used in different optimization problems.

This section is divided as follows: Section 3.3.1 is dedicated to show the new velocity update equation of ISAPSO algorithm; differences between SAPSO and ISAPSO algorithms are discussed in Section 3.3.2; Section 3.3.3 describes a step by step mathematical proof about the property rotation invariance; and Section 3.3.4 provides the minimum number of executions required to achieve reliable results based on the method described earlier in Section 3.2.3.

3.3.1 The velocity update equation

As expected, ISAPSO algorithm is mainly based on SAPSO algorithm's velocity equation. However, ISAPSO algorithm introduces a rotation matrix $\mathbf{W}$ which is built

by the exponential map described in Section 2.9.5. The exponential map brings a fast alternative to build a rotation matrix for higher dimensions. On the other hand, the rotation matrix is not exact and the final vector is approximated with the corresponding rotation angle, as well as the magnitude of the vector is modified.

The velocity update equation of ISAPSO algorithm is defined as follows

$$\vec{\mathbf{v}}_i^{t+1} = w^t \vec{\mathbf{v}}_i^t + dir \times \left[\vec{\mathbf{\Omega}}_i^t + \vec{\mathbf{\Upsilon}}_i^t\right], \tag{3.14}$$

where

$$\vec{\mathbf{\Omega}}_i^t = \begin{cases} I_i^t c_1 \phi_1 \left(\vec{\mathbf{g}}^t - \vec{\mathbf{x}}_i^t\right) & \text{if } dir = 1, \\ I_i^t c_1 \phi_1 \dot{\mathbf{W}}_i^t \left(\vec{\mathbf{g}}^t - \vec{\mathbf{x}}_i^t\right) & \text{else} \end{cases} \tag{3.15}$$

and

$$\vec{\mathbf{\Upsilon}}_i^t = \begin{cases} \left(I_i^t - 1\right) c_2 \phi_2 \nabla f(\vec{\mathbf{x}}_i^t) & \text{if } dir = 1, \\ \left(I_i^t - 1\right) c_2 \phi_2 \ddot{\mathbf{W}}_i^t \nabla f(\vec{\mathbf{x}}_i^t) & \text{else} \end{cases} \tag{3.16}$$

The main mechanism about attraction and repulsion controlled by dir variable is borrowed from SAPSO algorithm, but now the rotation matrix $\mathbf{W}_i^t$ introduces a small perturbation in the magnitude and direction of the vectors $(\vec{\mathbf{g}}^t - \vec{\mathbf{x}}_i^t)$ and $\nabla f(\vec{\mathbf{x}}_i^t)$ in the repulsion phase only. It is important to note in Equations 3.15 and 3.16 that the perturbation in the direction of the best global memory and the gradient vector is only used when the swarm is in the repulsion phase ($dir = -1$), whereas the attraction phase has no rotation matrix. As a consequence, when the swarm is in the repulsion phase, a small angle divergence is introduced no matter $I_i^t = 0$ or $I_i^t = 1$. One may expect to give the particle a chance to investigate other areas of the search domain.

ISAPSO algorithm still uses the social-only model described in (KENNEDY, 1997a) and the cognitive term is substituted by the gradient component. This is rather preferred as the gradient information brings a deterministic characteristic to the algorithm, which may decreases the random walk influence inherent to any metaheuristic approach. The decreasing of random walk influence is desired in situations where a basin of attraction is discovered by a particle, i.e., instead of flying around the discovered area governed by biased random movements, the particle would deterministically follows the negative direction of the gradient. Thus, much less time would be spent to examine this region.

3.3.2 Relations between SAPSO and ISAPSO algorithms

There are essential differences between both algorithms. Besides the fundamental idea of development a new version based on finding better solutions and low computational time, ISAPSO algorithm provides valuable difference against SAPSO one. The major differences and relations are highlighted as follows:

- Unlike the SAPSO algorithm, there are no heuristics to clamp velocities, gradient vectors, or bound the search space. ISAPSO algorithm simply compresses the searching inside the initial search domain and no bound handling is used throughout the search process.

- In the attraction phase, ISAPSO algorithm is strictly rotationally invariant, while in the repulsion phase, ISAPSO algorithm is rotationally invariant in a stochastic sense, i.e., the algorithm has directional diversity by applying a small perturbation in the vector direction of global memory and gradient information, which is rather preferable in contrast of the way SAPSO algorithm works.

- The computational time was fastened by performing partial derivatives with forward-differencing only when $I_i^t = 0$, whereas SAPSO algorithm always performs this calculations.

- Both algorithms use diversity control. However, ISAPSO algorithm keeps track of when the swarm is in the attraction or repulsion phase. This tracking ensures the proper use of rotation matrix W_i^t and guarantees no disturbance of the gradient information when required.

- Also, both algorithms use the fast information exchange among particles, i.e., ith particle evaluates whether its current position is better than the global memory of the swarm, if so, it updates the global memory with ith particle's position. Then, $i + 1$th particle moves in the direction of the actual global memory.

Beyond the above relations between both algorithms, ISAPSO algorithm has the property *rotation invariance*, i.e., there is no change in the performance of the algorithm when the rotation of the coordinate system is applied, which inplies the algorithm has no biases in the direction of the coordinate system (SPEARS; GREEN; SPEARS, 2010). Furthermore, there is no extra burden to match the angle of bias with the angle of the coordinate system (WILKE; KOK; GROENWOLD, 2007a; WILKE; KOK; GROENWOLD, 2007b; BONYADI; MICHALEWICZ; LI, 2014; BONYADI; MICHALEWICZ, 2014; HARIYA; SHINDO; JIN'NO, 2016) in order to have a good performance. This also poses the algorithm to be applicable to a larger class of problems.

Although rotation invariance is desirable in this context, the authors of the work described in (WILKE; KOK; GROENWOLD, 2007a) showed this property lacks directional diversity when the particle is flying throughout the search space. They also presented that rotation invariance and directional diversity are not mutually exclusive properties. From this point of view, it is important to reestablish a directional diversity in the particle's movement at some extent. The rotation matrix is used to avoid the lacking of directional

diversity, but it also provides the property rotational invariance in a stochastic sense. The next section carries out the mathematical proof of this assumption.

3.3.3 Mathematical proof of rotationally invariant algorithm

This section provides a mathematical proof that ISAPSO algorithm is strictly rotationally invariant when the swarm is in the attraction phase, and rotationally invariant in a *stochastic sense* otherwise. The main focus is to satisfy Equations 2.24 and 2.28 previously discussed in Section 2.8. All mathematical deductions in this section have the intention to prove indifference in rotation of the coordinate system, which is the most important property of ISAPSO algorithm. First, it is considered a swarm of n particles in a d-dimensional search space. The velocity vector of the ith particle is obtained from the velocity update equation and it can be interpreted as two parts as follows

$$\vec{\mathbf{v}}_i^{t+1} = w\vec{\mathbf{v}}_i^t + \vec{\boldsymbol{\psi}}_i^t, \tag{3.17}$$

where w is the inertia coefficient, $\vec{\mathbf{v}}_i^t$ is the velocity vector of the ith particle at iteration t. This first part is considered *deterministic* at iteration t. The second part of the summation is defined as *stochastic* and it is represented by the stochastic vector $\vec{\boldsymbol{\psi}}_i^t$.

The stochastic vector of the ISAPSO version is given by

$$\vec{\boldsymbol{\psi}}_i^t = dir \times \left[\vec{\boldsymbol{\Omega}}_i^t + \vec{\boldsymbol{\Upsilon}}_i^t\right]. \tag{3.18}$$

Due to the binary variables dir and I_i^t, there are four possible cases in this stochastic vector:

- case 1: $dir = 1$ and $I_i^t = 1$;
- case 2: $dir = 1$ and $I_i^t = 0$;
- case 3: $dir = -1$ and $I_i^t = 1$;
- case 4: $dir = -1$ and $I_i^t = 0$.

In the following sections, each case is treated separately according to the state of variables dir and I_i^t. Note that, the following mathematical proof is determined to prove rotation invariance only. Thus, no concerning about scale and translation is observed.

3.3.3.1 Case 1: $dir = 1$ and $I_i^t = 1$

In this case, the stochastic vector is reduced to

$$\vec{\boldsymbol{\psi}}_i^t = \vec{\boldsymbol{\Omega}}_i^t = c_1\phi_1\left(\vec{\mathbf{g}}^t - \vec{\mathbf{x}}_i^t\right). \tag{3.19}$$

From Section 2.8, the transformed stochastic velocity vector is derived by

$$\begin{aligned}\hat{\vec{\boldsymbol{\psi}}}_i^t &= c_1\phi_1\left(\hat{\vec{\mathbf{g}}}^t - \hat{\vec{\mathbf{x}}}_i^t\right)\\ &= c_1\phi_1\left(Q\vec{\mathbf{g}}^t - Q\vec{\mathbf{x}}_i^t\right)\\ &= Q\left(c_1\phi_1\left(\vec{\mathbf{g}}^t - \vec{\mathbf{x}}_i^t\right)\right)\\ &= Q\vec{\boldsymbol{\psi}}_i^t.\end{aligned} \tag{3.20}$$

In sequence, the transformed position update equation for $\hat{\vec{\mathbf{x}}}_i^t$ is given by

$$\begin{aligned}\hat{\vec{\mathbf{x}}}_i^{t+1} &= \hat{\vec{\mathbf{x}}}_i^t + w\hat{\vec{\mathbf{v}}}_i^t + \hat{\vec{\boldsymbol{\psi}}}_i^t\\ &= Q\vec{\mathbf{x}}_i^t + Qw\vec{\mathbf{v}}_i^t + Q\vec{\boldsymbol{\psi}}_i^t\\ &= Q\left(\vec{\mathbf{x}}_i^t + w\vec{\mathbf{v}}_i^t + \vec{\boldsymbol{\psi}}_i^t\right)\\ &= Q\vec{\mathbf{x}}_i^{t+1}.\end{aligned} \tag{3.21}$$

The deductions presented in Equations 3.20 and 3.21 satisfy both Equations 2.24 and 2.28. Therefore, case (1) can be considered strictly rotationally invariant, i.e., when the swarm is in the attraction phase ($dir = 1$) and the global tendency of the swarm is preferred ($I_i^t = 1$).

3.3.3.2 Case 2: $dir = 1$ and $I_i^t = 0$

Now, considering case (2) where $dir = 1$ and $I_i^t = 0$, the stochastic velocity vector is given by

$$\vec{\boldsymbol{\psi}}_i^t = \vec{\boldsymbol{\Upsilon}}_i^t = -c_2\phi_2\nabla f(\vec{\mathbf{x}}_i^t). \tag{3.22}$$

The transformed stochastic velocity vector $\hat{\vec{\boldsymbol{\psi}}}_i^t$ requires $\nabla f(\hat{\vec{\mathbf{x}}}_i^t)$, but $\hat{\vec{\mathbf{x}}}_i^t$ in this context is the same as

$$\hat{\vec{\mathbf{x}}}_i^t = Q\vec{\mathbf{x}}_i^t \quad \Leftrightarrow \quad \vec{\mathbf{x}}_i^t = Q^{-1}\hat{\vec{\mathbf{x}}}_i^t \tag{3.23}$$

Thus, optimizing deterministically $\nabla f(\vec{\mathbf{x}}_i^t)$ is equivalent to optimize $\nabla g(\hat{\vec{\mathbf{x}}}_i^t)$ as follows

$$\begin{aligned}g(\hat{\vec{\mathbf{x}}}_i^t) &= f(Q^{-1}\hat{\vec{\mathbf{x}}}_i^t)\\ \nabla g(\hat{\vec{\mathbf{x}}}_i^t) &= Q^{-T}\nabla f(Q^{-1}\hat{\vec{\mathbf{x}}}_i^t).\end{aligned} \tag{3.24}$$

In the following, the transformed stochastic velocity vector is derived by

$$\begin{aligned}\hat{\vec{\boldsymbol{\psi}}}_i^t &= -c_2\phi_2\nabla g(\hat{\vec{\mathbf{x}}}_i^t)\\ &= -c_2\phi_2 Q^{-T}\nabla f(Q^{-1}\hat{\vec{\mathbf{x}}}_i^t)\\ &= Q^{-T}\Big(-c_2\phi_2\nabla f(\vec{\mathbf{x}}_i^t)\Big)\\ &= Q^{-T}\vec{\boldsymbol{\psi}}_i^t\\ &= (Q^{-1})^T\vec{\boldsymbol{\psi}}_i^t\\ &= Q\vec{\boldsymbol{\psi}}_i^t.\end{aligned} \tag{3.25}$$

Finally, the transformed position update equation for $\hat{\vec{\mathbf{x}}}_i^t$ is given by

$$\begin{aligned}\hat{\vec{\mathbf{x}}}_i^{t+1} &= \hat{\vec{\mathbf{x}}}_i^t + w\hat{\vec{\mathbf{v}}}_i^t + \hat{\vec{\boldsymbol{\psi}}}_i^t\\ &= Q\vec{\mathbf{x}}_i^t + Qw\vec{\mathbf{v}}_i^t + Q\vec{\boldsymbol{\psi}}_i^t\\ &= Q\Big(\vec{\mathbf{x}}_i^t + w\vec{\mathbf{v}}_i^t + \vec{\boldsymbol{\psi}}_i^t\Big)\\ &= Q\vec{\mathbf{x}}_i^{t+1}.\end{aligned} \tag{3.26}$$

In this form, ISAPSO algorithm is also strictly rotationally invariant in case (2) when $dir = 1$ and $I_i^t = 0$. Moreover, ISAPSO algorithm is strictly rotationally invariant in the attraction phase, independently of the state I_i^t be 0 or 1.

3.3.3.3 Case 3: $dir = -1$ and $I_i^t = 1$

The stochastic vector is reduced to

$$\vec{\boldsymbol{\psi}}_i^t = \vec{\boldsymbol{\Omega}}_i^t = -c_1\phi_1\dot{\mathbf{W}}_i^t\Big(\vec{\mathbf{g}}^t - \vec{\mathbf{x}}_i^t\Big). \tag{3.27}$$

From Section 2.8, the transformed stochastic velocity vector is derived by

$$\begin{aligned}\hat{\vec{\boldsymbol{\psi}}}_i^t &= -c_1\phi_1\hat{\dot{\mathbf{W}}}_i^t\Big(\hat{\vec{\mathbf{g}}}^t - \hat{\vec{\mathbf{x}}}_i^t\Big)\\ &= -c_1\phi_1\hat{\dot{\mathbf{W}}}_i^t\Big(Q\vec{\mathbf{g}}^t - Q\vec{\mathbf{x}}_i^t\Big)\\ &= -c_1\phi_1\hat{\dot{\mathbf{W}}}_i^t Q\Big(\vec{\mathbf{g}}^t - \vec{\mathbf{x}}_i^t\Big).\end{aligned} \tag{3.28}$$

Note that the required transformation between $\hat{\vec{\boldsymbol{\psi}}}_i^t$ and $\vec{\boldsymbol{\psi}}_i^t$ is given by

$$\begin{aligned}\hat{\vec{\boldsymbol{\psi}}}_i^t &= Q\vec{\boldsymbol{\psi}}_i^t\\ &= Q\Big(-c_1\phi_1\dot{\mathbf{W}}_i^t\Big(\vec{\mathbf{g}}^t - \vec{\mathbf{x}}_i^t\Big)\Big)\\ &= -c_1\phi_1 Q\dot{\mathbf{W}}_i^t\Big(\vec{\mathbf{g}}^t - \vec{\mathbf{x}}_i^t\Big).\end{aligned} \tag{3.29}$$

Since the relation between $\hat{\vec{\boldsymbol{\psi}}}_l$ and ψ_l has to hold $\forall Q \in \mathrm{Orth}^+$, from Section 2.8.2 the solution is

$$\hat{\dot{\mathbf{W}}}_i^t = \dot{\mathbf{W}}_i^t = a_l I_i^t \quad \text{for } l = 1, 2. \tag{3.30}$$

Exactly as before, a strict enforcement of rotational invariance reduces the ISAPSO velocity update equation to the version presented in Section 2.8.1. In contrast, since the velocity equation is stochastic, it is possible to satisfy Equation 3.30 in an average sense only, letting the rotation matrix $\mathbf{W}_i^t$ provides small rotations.

In sequence, the transformed position update equation for $\hat{\vec{\mathbf{x}}}_i^t$ is given by

$$\begin{aligned} \hat{\vec{\mathbf{x}}}_i^{t+1} &= \hat{\vec{\mathbf{x}}}_i^t + w\hat{\vec{\mathbf{v}}}_i^t + \hat{\vec{\boldsymbol{\psi}}}_i^t \\ &= Q\vec{\mathbf{x}}_i^t + Qw\vec{\mathbf{v}}_i^t - c_1\phi_1\dot{\mathbf{W}}_i^t Q\Big(\vec{\mathbf{g}}^t - \vec{\mathbf{x}}_i^t\Big). \end{aligned} \tag{3.31}$$

By considering ISAPSO algorithm invariant under rotation, let assume $Q = I_i^t$ and assume the random matrices $\hat{\dot{\mathbf{W}}}_i^t = \dot{\mathbf{W}}_i^t$ are generated in the same manner, then

$$\begin{aligned} \hat{\vec{\mathbf{x}}}_i^{t+1} &= Q\Big(\vec{\mathbf{x}}_i^t + w\vec{\mathbf{v}}_i^t - c_1\phi_1\dot{\mathbf{W}}_i^t\Big(\vec{\mathbf{g}}^t - \vec{\mathbf{x}}_i^t\Big)\Big) \\ &= Q\vec{\mathbf{x}}_i^{t+1}. \end{aligned} \tag{3.32}$$

As a result, the ISAPSO velocity update equation is rotationally invariant in a stochastic sense in case (3), since a small perturbation in the direction of social and gradient components is considered by the rotation matrix $\mathbf{W}$.

3.3.3.4 Case 4: $dir = -1$ and $I_i^t = 0$

In the last case (4), $dir = -1$ and $I_i^t = 0$, the stochastic velocity vector is given by

$$\vec{\boldsymbol{\psi}}_i^t = -c_2\phi_2\hat{\dot{\mathbf{W}}}_i^t\nabla f(\vec{\mathbf{x}}_i^t). \tag{3.33}$$

Then, the transformed stochastic velocity vector is derived by

$$\begin{aligned} \hat{\vec{\boldsymbol{\psi}}}_i^t &= -c_2\phi_2\hat{\dot{\mathbf{W}}}_i^t\nabla g(\hat{\vec{\mathbf{x}}}_i^t) \\ &= -c_2\phi_2\hat{\dot{\mathbf{W}}}_i^t Q^{-T}\nabla f(Q^{-1}\hat{\vec{\mathbf{x}}}_i^t) \\ &= -c_2\phi_2\hat{\dot{\mathbf{W}}}_i^t Q\nabla f(\vec{\mathbf{x}}_i^t). \end{aligned} \tag{3.34}$$

Since the relation between $\hat{\vec{\boldsymbol{\psi}}}_l$ and $\vec{\boldsymbol{\psi}}_l$ has to hold $\forall Q \in \mathrm{Orth}^+$, according to Section 2.8.2 the solution is

$$\hat{\ddot{\mathbf{W}}}_i^t = \ddot{\mathbf{W}}_i^t = a_l I \quad \text{for } l = 1, 2. \tag{3.35}$$

Finally, assuming $Q = I$ and $\hat{\ddot{\mathbf{W}}}_i^t = \ddot{\mathbf{W}}_i^t$ are generated equally, the transformed position update equation for $\hat{\vec{\mathbf{x}}}_i^t$ is given by

$$\begin{aligned}\hat{\vec{\mathbf{x}}}_i^{t+1} &= \hat{\vec{\mathbf{x}}}_i^t + w\hat{\vec{\mathbf{v}}}_i^t + \hat{\vec{\boldsymbol{\psi}}}_i^t \\ &= Q\vec{\mathbf{x}}_i^t + Qw\vec{\mathbf{v}}_i^t - c_2\phi_2\ddot{\mathbf{W}}_i^t Q\nabla f(\vec{\mathbf{x}}_i^t) \\ &= Q\left(\vec{\mathbf{x}}_i^t + w\vec{\mathbf{v}}_i^t - c_2\phi_2\ddot{\mathbf{W}}_i^t\nabla f(\vec{\mathbf{x}}^t)\right) \\ &= Q\vec{\mathbf{x}}_i^{t+1}.\end{aligned} \tag{3.36}$$

In summary, ISAPSO algorithm is strictly rotationally invariant in a stochastic sense in cases (1) and (2), and rotationally invariant in stochastic sense in cases (3) and (4). Cases (1) and (2) happen when the swarm is in the attraction phase ($dir = 1$), and cases (3) and (4) occur when the swarm is in the repulsion phase. As previously stated, this behavior poses a possibility to find a new local minima (maybe global optimum). Additionally, the same local optimum can be investigated through a different angle, since small perturbations are propagated in cases (3) and (4).

3.3.4 Number of executions based on the proposed method

The experimental results performed with ISAPSO algorithm is later examined in Section 4.3. However, before proceeding with the next chapter, this section reports the minimum number of experiments that will be used in the following chapter. The formal definition of this method was explained in Section 3.2.3.

Following the same experiments previously viewed in Section 3.2.3, the method selects the number of experiments required to stabilize the curves of success rate for each optimization problem. The test functions are derived from CEC 2017 benchmark optimization problems described later in Section 4.2.1. Table 3 presents the parameters used in this simulation.

In Figure 13, two outcomes from the method are depicted. Figure 13a shows all success rate curves considering the maximum number of experiments defined as $E^\star = 500$. One may note the higher the number of experiments, the more stable the curves become. For each success rate curve, a number of experiment is extracted, and the average of the number of experiments is considered in this scenario.

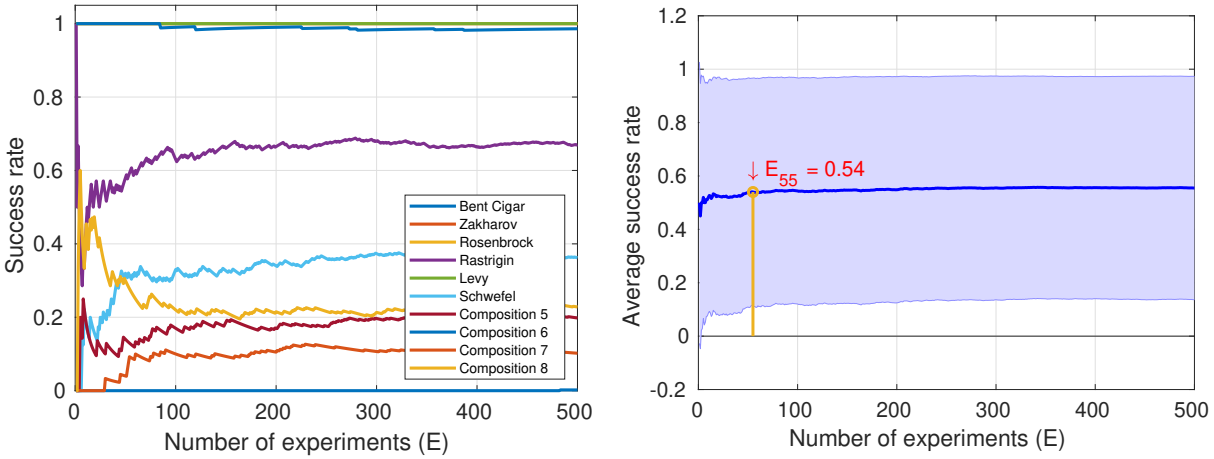
As a matter of visualization, Figure 13b depicts the average success rate by considering each of ten test functions and it highlights the minimum number of experiments. Thus, in the result section, the numerical simulations with ISAPSO algorithm will consider $E = 55$ in all experiments.

Table 3 – ISAPSO parameters to find the minimum number of executions.

Variable	Conf.
Number of particles (n)	20
Number of dimensions (d)	2
Social coefficient (c_1)	1.4962
Inertia weight (w)	0.7298
Number of iterations (T)	1000
Number of experiments (E)	500
Sequence evaluation ($cMax$)	3
Stop criterion (ϵ)	10^{-2}
Deviation angle (α)	3°

Source: produced by the author.

Figure 13 – Number of experiments based on CEC 2017 optimization problems using ISAPSO algorithm with fast information exchange among particles.



(a) Success rate in each function.

(b) Average number of experiments.

Source: produced by the author.

4 Experimental results and discussions

This chapter details the numerical simulations and experimental results carried out in this thesis. The results are divided in three sections through a chronological order of study. A brief description of each section is given below:

- Section 4.1: the first experiment evaluates SAPSO algorithm on the ability of finding the global minimum on De Jong's benchmark functions with the number of dimensions set as 2, 10, 20, 30, 60, 80, 100, 120, and 150, except for the bi-dimensional ones. After that, SAPSO algorithm is compared with the most competitive PSO algorithm, i.e., the one with acceptable results in the first experiment. Both algorithms are tested in multimodal functions with even higher dimensions. The computational times required to execute the algorithms are provided based on the average of $E = 20$ runs. A discussion about the algorithms' reliability related to the success rate and the number of iterations required to reach a feasible solution is also provided. Finally, an analysis of the diversity control is outlined between the SAPSO algorithm and the PSO algorithms under evaluation. In addition, a relation between the average number of times a repulsion phase occurs and the average number of iterations required to achieve the global minimum is described.

- Section 4.2: an empirical analysis of rotation and information exchange among particles is supplied in this section. First, CEC 2017 benchmark suite problems is summarized. Later, numerical simulation results reports the performance of each PSO versions under evaluation. All simulation results are based on the minimum number of executions $E = 134$ found by the proposed algorithm describe in Section 3.2.3. Further, two well-known statistical hypothesis tests are used to evaluate whether statistical significance in the results is noted.

- Section 4.3: based on both previous enhancements and developments, ISAPSO algorithm is detailed in this section. The methodology of evaluation of the new algorithm is similar to the ones presented above. The results are averaged over $E = 55$ executions. ISAPSO algorithm is compared in terms of performance against seven related PSO versions, and a statistical hypothesis test is also reported.

All simulations were conducted in a PC described as follows: Operational System Ubuntu 18.04 LTS, CPU Intel® Core™ i7-7500U, and 8 GB of RAM. The MATLAB R2018a environment was the programming language used to code and test all algorithms described in this thesis.

4.1 Results: SAPSO algorithm

Firstly, before describing the simulations and results, the benchmark optimization problems used in the following experiments are discussed in terms of their definitions and search domains.

4.1.1 De Jong's benchmark problems

The test functions based on the De Jong's benchmark optimization problems (JONG, 1975) are exhibited in Table 4. Each function offers different challenges for any optimization algorithms, such as: many local minima surrounding global minimum, nearly flat regions with small gradient information, d-dimensional continuous space for searching, and decentralized global minimum. The functions are used to assert the qualities of the proposed algorithm, particularly on multimodal functions where the SAPSO algorithm must be able to avoid local minima and fine-tune the global optimum as a final result. In addition, the different challenges presented in each test function will be addressed by the proposed algorithm and the use of gradient-based information and diversity control mechanism will be under test of robustness, as well as the well-known test of finding the global optimum of each function.

The Sphere (Figure 14a) and Ellipsoid (Figure 14f) test functions are unimodal, d-dimensional quadratic function and convex, i.e., a single local minimum (which is the global minimum) is observed. The simplicity of f_1 and f_6 tests the algorithms' ability of finding the global minimum. One can consider these functions as a baseline test of any optimization algorithm. The Rosenbrock (Figure 14b) is another standard test function. f_2 is unimodal, non-convex and d-dimensional quadratic function with a single global minimum. This function tests the algorithms' ability of navigating flat areas of the search space with small gradient information. The Rastrigin (Figure 14c) and Griewank (Figure 14d) functions are variations of f_1 with addition of cosine modulation to produce many local minima. Thus, f_3 and f_4 are highly multimodal. The Alpine (Figure 14h) and Levi (Figure 14i) are complex multimodal function with lots of local minima around the global minimum. Functions f_8 and f_9 have the global minimum far from the center of the default hypercube usually used to evaluate a metaheuristic algorithm, which is interesting to test the algorithms' ability of finding a decentralized global minimum. The Ackley (Figure 14e) test function is widely used in the optimization literature. Function f_5 is characterized by a nearly flat outer region and a hole at the center. This function tests the algorithms' ability of avoiding local minima. The Schaffer N.2 (Figure 14g) and Levi N.13 (Figure 14j) are bi-dimensional, continuous and non-convex test functions. Functions f_7 and f_{10} are used to test the algorithms' ability of addressing low dimensional functions, which is suitable to evaluate the robustness of the algorithms under these conditions.

Table 4 – Test functions used for comparison of the algorithms.

Test function	Equation	Search domain
Sphere (f_1)	$\sum_{i=1}^{d} x_i^2$	$-100 \leq x_i \leq 100$
Rosenbrock (f_2)	$\sum_{i=1}^{d-1} 100\left(x_{i+1} - x_i^2\right)^2 + \left(1 - x_i\right)^2$	$-5.12 \leq x_i \leq 5.12$
Rastrigin (f_3)	$10d + \sum_{i=1}^{d} \left(x_i^2 - 10\cos\left(2\pi x_i\right)\right)$	$-5.12 \leq x_i \leq 5.12$
Griewank (f_4)	$1 + \frac{1}{4000}\sum_{i=1}^{d} x_i^2 - \prod_{i=1}^{d} \cos\left(\frac{x_i}{\sqrt{i}}\right)$	$-100 \leq x_i \leq 100$
Ackley (f_5)	$-20\exp\left(-0.2\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}\right)$ $-\exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos\left(2\pi x_i\right)\right) + 20 + \exp\left(1\right)$	$-50 \leq x_i \leq 50$
Ellipsoid (f_6)	$\sum_{i=1}^{d} ix^2$	$-5.12 \leq x_i \leq 5.12$
Schaffer N.2 (f_7)	$0.5 + \frac{\sin^2\left(x_1^2 - x_2^2\right) - 0.5}{\left(1 + 0.001\left(x_1^2 + x_2^2\right)\right)^2}$	$-100 \leq x_i \leq 100$
Alpine (f_8)	$\sum_{i=1}^{d} \lvert x_i \sin(x_i) + 0.1x_i \rvert$	$0 \leq x_i \leq 20$
Levi (f_9)	$\sin^2\left(\pi w_1\right) + \sum_{i=1}^{d-1}\left(w_i - 1\right)^2\left(1 + 10\sin^2\left(\pi w_i + 1\right)\right)$ $+w_d^2\left(1 + \sin^2\left(2\pi w_d\right)\right)$ where $w_i = 1 + \frac{x_i - 1}{4},\ \forall\ i = 1, \ldots, d$	$-10 \leq x_i \leq 10$
Levi N.13 (f_{10})	$\sin^2\left(3\pi x_1\right) + \left(x_1 - 1\right)^2\left(1 + \sin^2\left(3\pi x_2\right)\right)$ $+\left(x_2 - 1\right)^2\left(1 + \sin^2\left(2\pi x_2\right)\right)$	$-10 \leq x_i \leq 10$

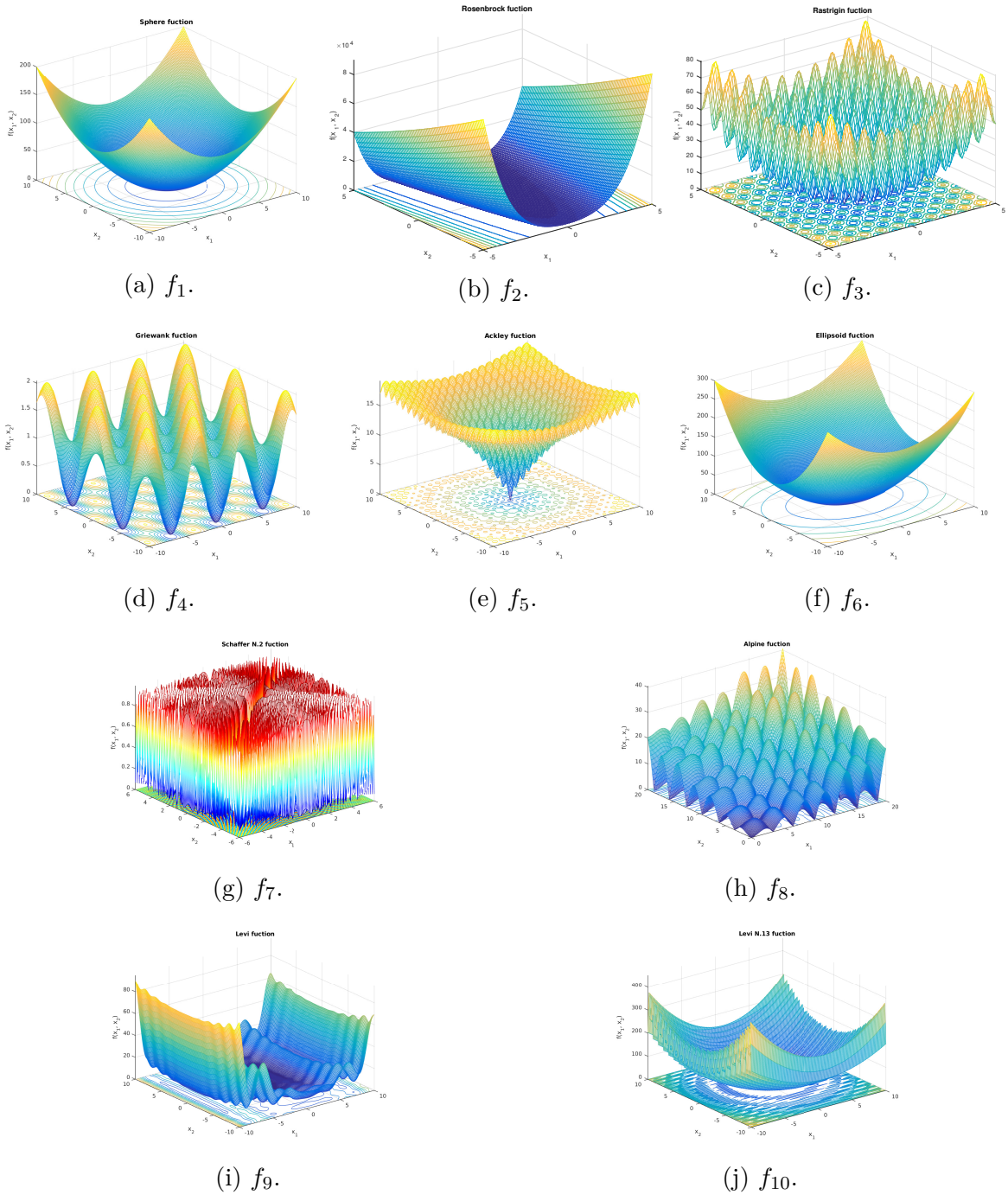
Source: produced by the author.

The test functions are continuous and bounded by a search domain in the form of hypercube. Each test function has only one global minimum with value equal to zero. The number of dimensions d is chosen according to the literature (VESTERSTRØM; RIGET, 2002; NOEL, 2012; HAN; LIU, 2014), which is common for comparisons among optimization algorithms.

4.1.2 Toward the global minimum

In this section, the proposed SAPSO algorithm is compared to ARPSO, GPSO and DGHPSOGS in terms of finding the global minimum in functions based on De Jong's benchmark optimization problems. In all experiments, the population size for all algorithms is 20. The cognitive and social coefficients of ARPSO, DGHPSOGS (attraction phase) and

Figure 14 – Bi-dimensional surface of De Jong's benchmark functions.



(a) f_1. (b) f_2. (c) f_3.

(d) f_4. (e) f_5. (f) f_6.

(g) f_7. (h) f_8.

(i) f_9. (j) f_{10}.

Source: produced by the author.

GPSO are all set as 2. The gradient and social coefficients of DGHPSOGS are both 2.1 when the algorithm is in the repulsion phase. The lower bound of ARPSO and DGHPSOGS are set as 10^{-6} and the upper bound of ARPSO is 0.25. Note that the parameters of each PSO were obtained in their original papers and can be found in Appendix B in the Table 27.

The following parameters of SAPSO are set based on empirical studies made by

the author of this thesis and inspired by works presented in (VESTERSTRØM; RIGET, 2002; NOEL, 2012; HAN; LIU, 2014; NOEL; JANNETT, 2004). In (HAN; LIU, 2014), the authors exhaustively test a range of c_1 and c_2, varying from 1 to 4. Herein, the author uses the same range for c_1 parameter, which means the importance given to the global search direction. However, note that the c_2 parameter must be very low, since it corresponds to the gradient information of the function in the current particle's position, varying from 10^{-5} to 10^{-1}. The SAPSO parameters: (F1, F6, F7, F9, F10) $c_1 = 2$ and $c_2 = 10^{-2}$; (F2) $c_1 = 2$ and $c_2 = 10^{-3}$; (F3) $c_1 = 3$ and $c_2 = 10^{-3}$; (F4) $c_1 = 3$ and $c_2 = 10^{-2}$; (F5, F8) $c_1 = 4$ and $c_2 = 10^{-1}$. Also based on (VESTERSTRØM; RIGET, 2002), the lower (d_{low}) and upper (d_{high}) bounds are the same of ARPSO, respectively, 10^{-6} and 0.25 for all test functions. The author found $cMax = 3$ a suitable value for all test functions. In all experiments, the maximum number of iterations is 5000 and the total number of executions is 20. The numerical results are shown in terms of mean best solution found in all executions.

Table 5 exhibits the mean best solution for all test functions using four PSOs. The proposed SAPSO algorithm found the global minimum (or very close to the global minimum) in all test functions. Even in Rosenbrock and Levi functions where the global minima were not exactly found, the results are close to the minimum, which can be considered the optimal solution depending on the relaxation of the optimization problem. In general, the SAPSO algorithm reports better convergence accuracy in terms of global minimum values than other PSOs in all cases, followed by GPSO, ARPSO and DGHPSOGS.

Since the algorithms converged to the exact result in most functions, the author decided to extract an analysis of convergence curves for the Rosenbrock and Levi functions, which are the ones that no exact result was found for all algorithms. Figures 15 and 16 show the average convergence curves of all PSOs on Rosenbrock and Levi functions, respectively, with 10, 20 and 30 dimensions.

Figure 15 shows that the proposed SAPSO algorithm clearly converges better than the other PSOs with 5000 iterations. Also, if the maximum number of iterations was not a stop criterion, the final results would certainly be even better, since the curves of the global best position seem to become better positioned during the search process. The ARPSO and DGHPSOGS algorithms suffer from fitness stagnation on Rosenbrock function after 2000 iteration, regardless of the number of dimensions. The GPSO algorithm suffers from the same effect during the search process, however the results are quite better than ARPSO and DGHPSOGS. The GPSO algorithm requires more iterations to fine-tune its global best position during the search process, and in the case of 10 and 30 dimensions, the algorithm stagnated its fitness value for a long period, whereas the proposed SAPSO algorithm is frequently improving its global best position due to the small gradient information of the Rosenbrock function and repulsion phases during the search process.

Table 5 – Mean best solution found in 20 runs.

Function	d	**ARPSO**	**GPSO**	**DGHPSOGS**	**SAPSO**
Sphere	10	0	0	0	**0**
	20	0	0	0	**0**
	30	0	0	0	**0**
Rosenbrock	10	2.2727	$2.1698 \cdot 10^{-5}$	1.5947	$\mathbf{7.5115 \cdot 10^{-11}}$
	20	17.7008	$3.7747 \cdot 10^{-5}$	14.1068	$\mathbf{4.9591 \cdot 10^{-10}}$
	30	28.1702	$3.7954 \cdot 10^{-5}$	23.9049	$\mathbf{4.173 \cdot 10^{-9}}$
Rastrigin	10	0	0	0	**0**
	20	1.4462	0	4.0295	**0**
	30	8.7842	2.6899	20.5457	**0**
Griewank	10	0	0	0	**0**
	20	0	0	0	**0**
	30	0	0	$1.2503 \cdot 10^{-2}$	**0**
Ackley	10	0	0	0	**0**
	20	0	0	0	**0**
	30	0	0	$9.3672 \cdot 10^{-7}$	**0**
Ellipsoid	10	0	0	0	**0**
	20	0.1214	0	$1.85 \cdot 10^{-10}$	**0**
	30	2.7233	0	$1.53 \cdot 10^{-7}$	**0**
Schaffer N.2	2	0	0	0	**0**
Alpine	10	0.0011	$1.0858 \cdot 10^{-5}$	$2.0650 \cdot 10^{-15}$	**0**
	20	2.4738	1.0512	$1.0057 \cdot 10^{-11}$	**0**
	30	10.5205	2.7824	0.2795	**0**
Levi	10	0.1629	$4.3924 \cdot 10^{-10}$	$1.2785 \cdot 10^{-11}$	$\mathbf{3.5689 \cdot 10^{-13}}$
	20	0.5336	0.9280	0.1612	**0.0939**
	30	0.7979	10.0399	2.4415	**0.35**
Levi N.13	2	0	$1.0799 \cdot 10^{-8}$	0	**0**

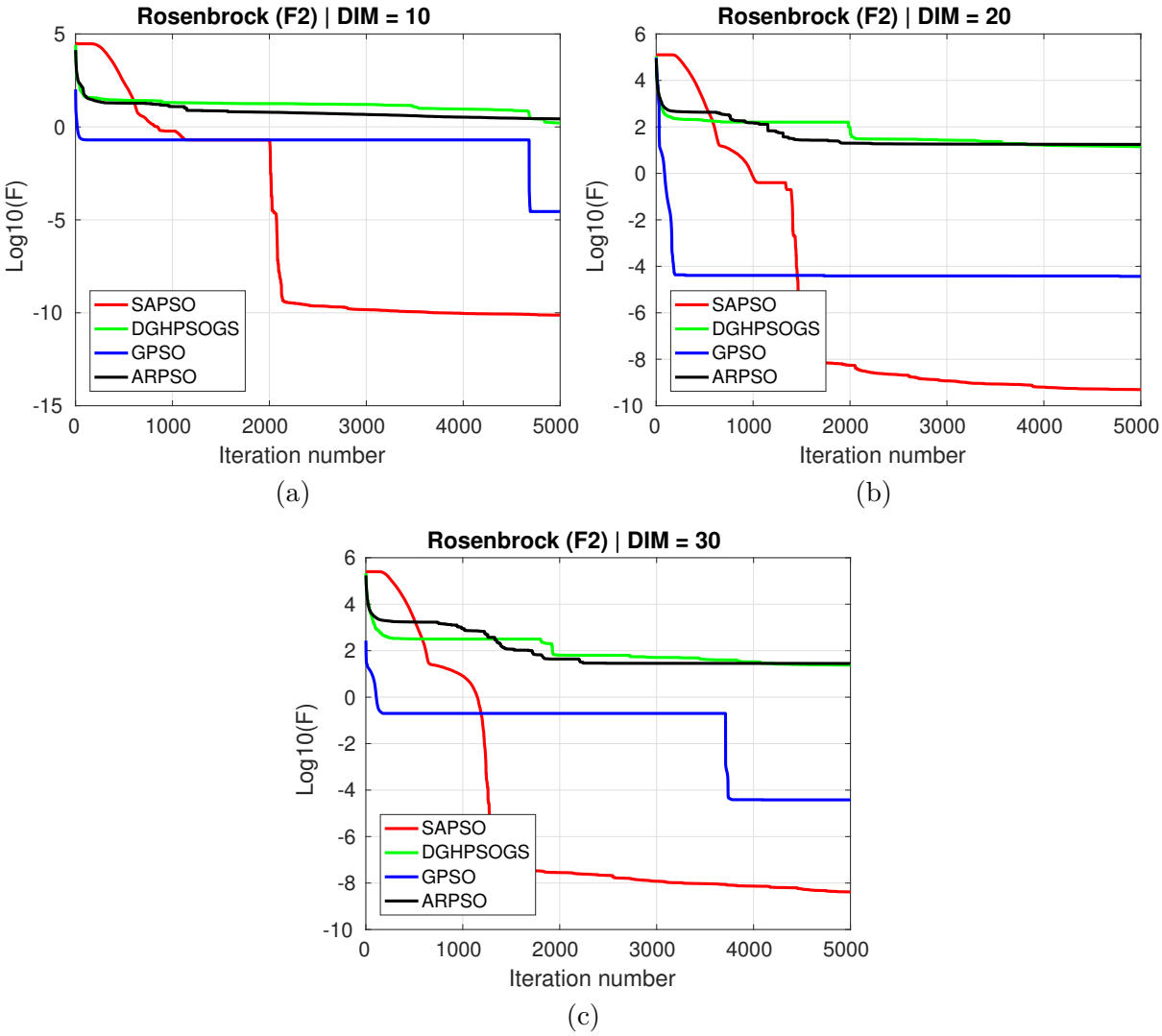
Source: produced by the author.

Similarly, Figure 16 presents the average convergence curves of the Levi function. One can note that the GPSO algorithm stagnates the global best position in the iteration 1000 for dimensions 10 and 30. A long period of fitness stagnation, from iteration 500 to 3500, is also noted when the dimension is 20. This seems to be caused by the lack of any mechanism of escaping from a local minima. Although the DGHPSOGS, ARPSO and SAPSO algorithms have the mechanism of attraction and repulsion, the proposed method avoided local minima during the search process, whereas the DGHPSOGS and ARPSO algorithms had some small periods of fitness stagnation during the search process.

4.1.3 Higher dimensional test functions

SAPSO algorithm had better performance when only the global best position was evaluated as the most important measure, followed by the GPSO algorithm, as seen in the

Figure 15 – Mean best solution versus iteration number using four PSOs: (a) Rosenbrock 10 dimensions; (b) Rosenbrock 20 dimensions; (c) Rosenbrock 30 dimensions.

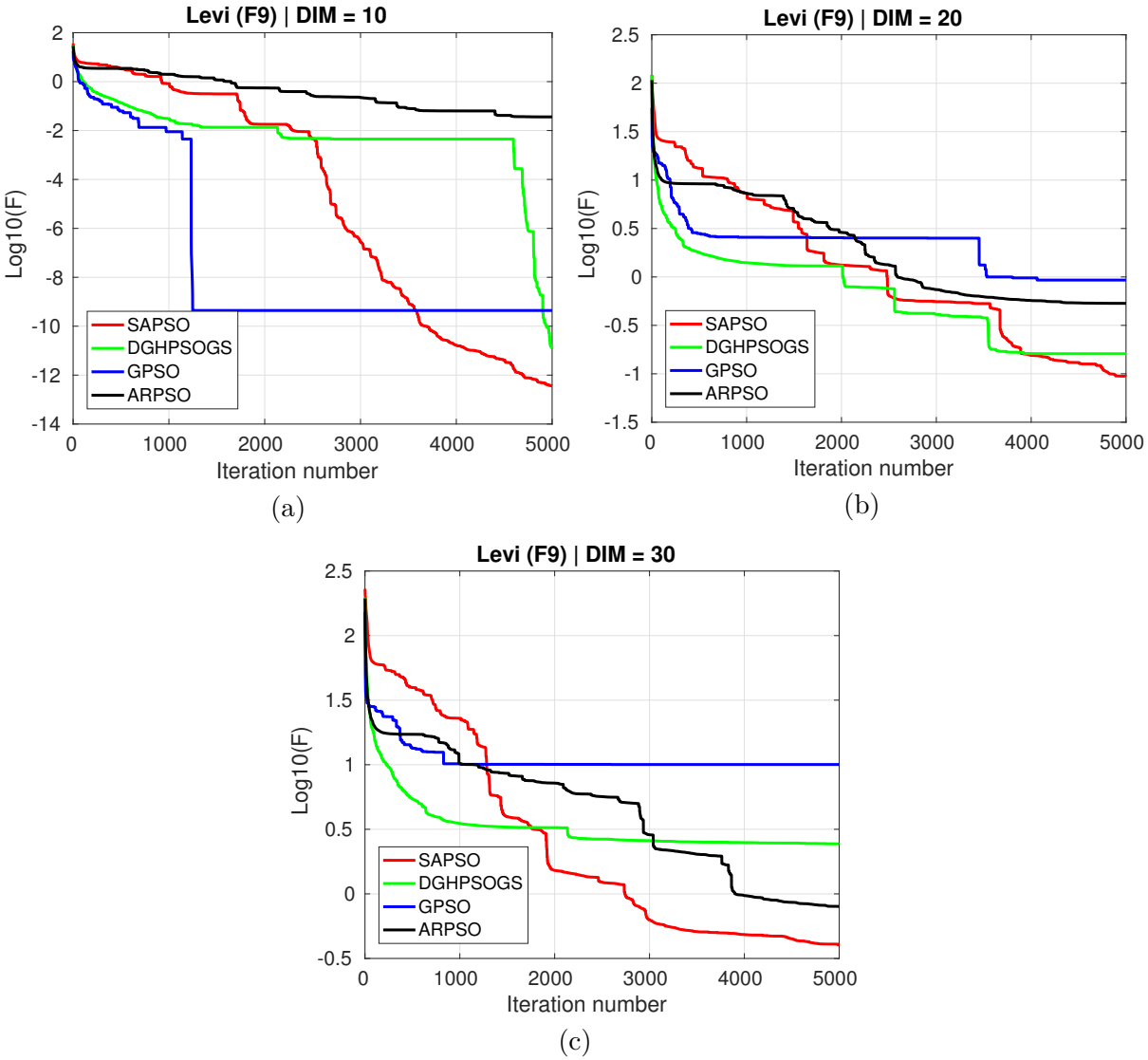


Source: produced by the author.

Table 5. With this in mind, the following experiments are executed in both algorithms only. The main goal of this experiment is to evaluate the robustness of the algorithms with even more difficult problems. One simple way to make the test functions more difficult is to elevate the number of dimensions.

Table 6 compares the average performances of SAPSO and GPSO only in multimodal test functions with even higher dimensions, from 60 to 150. The results show that SAPSO algorithm can maintain better approximations to the global minimum even in higher dimensions of the search space, whereas GPSO algorithm seems to get trapped into local minima of test functions, such as Rastrigin, Ackley, Alpine and Levi. The Griewank function has some different properties when is evaluated at higher dimensions. The work (LOCATELLI, 2003) presents a discussion about how this function behaves in higher

Figure 16 – Mean best solution versus iteration number using four PSOs: (a) Levi 10 dimensions; (b) Levi 20 dimensions; (c) Levi 30 dimensions.



Source: produced by the author.

dimensions. The author states that the function has a very large number of local minima, exponentially increasing with the number of dimensions. The fast increasing number of local minima suggests that the global minimum becomes extremely difficult to detect as dimension increases. Counter-intuitively, a consistent mathematical proof is given by the authors to prove the opposite, i.e., it becomes extremely easy to detect the global minimum when large number of dimensions is evaluated. When the dimension increases, the product of such cosine values becomes very small. Thus, the Griewank function, despite multiple local minima, behaves similar to a quadratic convex function. As presented in Table 6, the SAPSO algorithm ended up with near global minimum of the Griewank function which can be considered feasible in some engineering applications. As the GPSO algorithm uses a quasi Newton-Raphson method in the global best position at each iteration, the algorithm

Table 6 – Mean best solution found in 20 runs.

Function	d	**GPSO**	**SAPSO**
Griewank	60	**0**	0.0065
	80	**0**	0.0205
	100	**0**	0.016
	120	**0**	0.0532
	150	**0**	0.0287
Rastrigin	60	128.0997	**0**
	80	269.3824	**0**
	100	236.1511	**0**
	120	269.1834	**0**
	150	349.3266	**0**
Ackley	60	1	**0**
	80	0	**0**
	100	1.0225	**0**
	120	0	**0**
	150	0	**0**
Alpine	60	11.4362	**9.3852**
	80	57.1410	**23.3477**
	100	56.4144	**54.6302**
	120	106.6680	**63.0015**
	150	175.7591	**106.2794**
Levi	60	37.7036	**1.6057**
	80	54.7733	**3.0499**
	100	58.0076	**4.2114**
	120	62.9311	**5.8631**
	150	101.4227	**8.1463**

Source: produced by the author.

consequently finds the exact global minimum.

SAPSO algorithm reached exact results by finding the global minimum of the Rastrigin function for all dimensional settings. Although this function is a fairly difficult problem due to the large number of local minima, the mechanism of attraction and repulsion along with gradient information led the SAPSO algorithm to the global minimum position. On the other hand, the GPSO algorithm finished with poor results on Rastrigin function, mainly due to the lack of mechanism of attraction and repulsion, which is very difficult to escape from local minima and to properly explore the search space.

Although the Ackley function is highly multimodal, the SAPSO algorithm reaches the global minimum in all dimensional settings. The same principles to face the Rastrigin function is applied to face Ackley function. These types of multimodal functions must be faced with gradient information to improve the exploitation of valleys and other mechanism of attraction and repulsion the particles of the swarm, preventing the premature convergence

and avoiding computational efforts to lose many iterations by investigating the same area of the search space. On the other hand, the GPSO algorithm found the exact global minimum when the number of dimensions in the test function is set as 80, 120 and 150 but also found some difficulties when is set as 60 and 100, which is justifiable by some individual executions where the final result (the global best particle's fitness) is far from the true global minimum value.

The Alpine and Levi functions are highly multimodal and the global minimum is not centralized in the search space domain. As the particles are initialize randomly in the search space using a uniform distribution function, these types of functions bring new challenges to the algorithm, reducing the probability of an algorithm find the global minimum merely by luck. The PSO algorithms need to cleverly look for the global minimum and avoid all the local minima during the search space. As can be seen on Table 6, as the dimensions of Alpine and Levi functions increase, the level of difficulty in finding the exact global minimum becomes more challenging. The results of SAPSO algorithm are nearer the global minimum values than the ones obtained by the GPSO algorithm. As an overall analysis, the GPSO algorithm gathered poor results when the number of dimensions is higher than the default dimensional settings with 10, 20, 30. The most likely reason is that GPSO algorithm could not avoid local minima during the search process.

4.1.4 Computational time analysis and algorithm reliability

This section presents the computational time of each PSO algorithm and provides a discussion about the reliability of the proposed SAPSO algorithm. The results shown in the Table 7 are based on the average CPU time, regarding 20 runs of each algorithm in each test function configured for the dimensions 10, 20, and 30 (except for bi-dimensional functions). The table also presents the successful ratio and the average number of iterations required to reach a feasible solution in each test function. For this experiment, a feasible result is considered an error lower than 10^{-10} distant from the exact global minimum.

The proposed approach generally offers the highest reliability averaged over all functions, reaching feasible solutions with a successful ratio of 100% on all functions, except on Alpine and Levi functions. In higher dimensions such as 20 and 30 dimensions, the test functions are difficult to optimize, i.e., find the exact global minimum in the search domain of continuous variable is quite challenging. Note that, for instance, the 30 dimension Levi function, none of the approaches reached the stop criterion, but the SAPSO algorithm approximated the global minimum as could be seen in the previous Table 5.

Table 7 – Convergence speed based on average CPU time, ratio of success and average number of iterations required to converge in 20 runs.

Function	d	**Criterion**	**ARPSO**	**GPSO**	**DGHPSOGS**	**SAPSO**
Sphere	10	Time (sec)	2.546	**0.0408**	0.6652	3.1772
		Ratio (%)	**100**	**100**	**100**	**100**
		Iter.	1095.15	**14.2**	94	462.8
	20	Time (sec)	4.2586	**0.0594**	3.3571	6.7399
		Ratio (%)	95	**100**	**100**	**100**
		Iter.	1797.05	**20.85**	314.8	639
	30	Time (sec)	5.3245	**0.0785**	12.7193	10.0624
		Ratio (%)	95	**100**	95	**100**
		Iter.	2213.15	**23.9**	896.05	711.4
Rosenbrock	10	Time (sec)	12.7722	22.5368	46.991	**4.2591**
		Ratio (%)	0	0	0	**100**
		Iter.	5000	5000	5000	**428**
	20	Time (sec)	13.1014	34.4589	77.6425	**12.8736**
		Ratio (%)	0	0	0	**100**
		Iter.	5000	5000	5000	**815.8**
	30	Time (sec)	**13.2463**	41.5576	107.101	39.1361
		Ratio (%)	0	0	0	**100**
		Iter.	5000	5000	5000	**1802.8**
Rastrigin	10	Time (sec)	3.4878	1.3758	5.8281	**1.2901**
		Ratio (%)	**100**	95	95	**100**
		Iter.	1504.45	314.1	806.3	**178.6**
	20	Time (sec)	6.2266	16.5211	33.7384	**3.5005**
		Ratio (%)	85	40	55	**100**
		Iter.	2618.95	3054.05	2972.95	**309**
	30	Time (sec)	8.2892	24.0261	70.293	**5.2761**
		Ratio (%)	65	30	20	**100**
		Iter.	3456.9	3561.75	4554.9	**332.2**
Griewank	10	Time (sec)	2.9632	**0.106**	0.6406	0.54917
		Ratio (%)	**100**	**100**	**100**	**100**
		Iter.	1169.95	**21.6**	68.55	59.2
	20	Time (sec)	4.421	**0.0869**	8.6836	4.3347
		Ratio (%)	**100**	**100**	95	**100**
		Iter.	1692.6	**20.1**	575.2	285.8

Continued on next page

Function	d	Criterion	ARPSO	GPSO	DGHPSOGS	SAPSO
		Time (sec)	5.479	**0.0562**	36.9124	35.474
	30	Ratio (%)	90	**100**	70	**100**
		Iter.	2076.65	**12.65**	1770.55	1755.6
		Time (sec)	2.249	**0.1703**	0.9022	0.3053
	10	Ratio (%)	**100**	**100**	**100**	**100**
		Iter.	944.5	**34.85**	112.55	37.4
		Time (sec)	2.8142	**0.3983**	13.4706	8.1253
Ackley	20	Ratio (%)	**100**	**100**	**100**	**100**
		Iter.	1150.95	**58.55**	1058.9	633.6
		Time (sec)	3.8729	**1.575**	28.4259	17.6396
	30	Ratio (%)	**100**	**100**	80	**100**
		Iter.	1565.2	**195.5**	1624.35	1013.6
		Time (sec)	2.5814	**0.0556**	0.9925	1.2563
	10	Ratio (%)	**100**	**100**	**100**	**100**
		Iter.	1069.4	**18.85**	123.2	156.2
		Time (sec)	5.7766	**0.0905**	35.8376	2.2636
Ellipsoid	20	Ratio (%)	85	**100**	55	**100**
		Iter.	2338.45	**24.9**	2804.25	179
		Time (sec)	6.1336	**0.1177**	47.3343	3.0102
	30	Ratio (%)	80	**100**	55	**100**
		Iter.	2396.85	**26.1**	2777.15	175.2
		Time (sec)	0.1988	0.0192	0.0348	**0.0181**
Schaffer N.2	2	Ratio (%)	**100**	**100**	**100**	**100**
		Iter.	89.2	5.9	9.2	**4.8**
		Time (sec)	1.6122	7.5985	1.5072	**0.0517**
	10	Ratio (%)	**100**	90	**100**	**100**
		Iter.	693.65	512.9	212	**6.6**
		Time (sec)	**8.657**	31.6253	18.6761	23.2632
Alpine	20	Ratio (%)	40	45	**85**	**85**
		Iter.	3584.3	2275.5	1675.9	**2082.4**
		Time (sec)	**10.9362**	54.4005	46.1025	53.5494
	30	Ratio (%)	15	40	50	**80**
		Iter.	4484.15	3400.85	**3019.75**	3488
		Time (sec)	**11.1767**	13.2952	24.0394	18.6952
	10	Ratio (%)	40	0	90	**100**
		Iter.	4637.6	5000	3038.75	**2189.8**

Continued on next page

Levi

Function	*d*	**Criterion**	**ARPSO**	**GPSO**	**DGHPSOGS**	**SAPSO**
	20	Time (sec)	**12.4584**	18.7378	64.1383	62.2445
		Ratio (%)	0	0	10	**40**
		Iter.	5000	5000	4961.1	**4542**
	30	Time (sec)	**12.6167**	27.6092	89.8164	89.9835
		Ratio (%)	0	0	0	0
		Iter.	**5000**	**5000**	**5000**	**5000**
Levi N.13	2	Time (sec)	0.3927	6.9263	3.2726	**0.3394**
		Ratio (%)	**100**	25	**100**	**100**
		Iter.	132.95	3785.25	896.45	**95.15**

Source: produced by the author.

According to the "no free lunch theorem" (WOLPERT; MACREADY, 1997), one algorithm maintains an average performance on every aspect close to any other when considering a large number of test functions. In other words, an algorithm can not offer better performance on all kind of problems. The above phenomenon is also observed in this experiment. As presented in the summarized Table 8, regarding the average CPU time for each number of dimension, one can see that the SAPSO algorithm is the fastest approach up to dimension 10. Although the proposed approach is not the fastest approach among the PSO algorithms regarding the higher dimensions (but it is not the worst either). In this case, note that the gradient-based approaches take much time on higher dimensions mainly due to the time required to calculate the partial derivatives. Indeed, this influences the time required to run the algorithm and is also observed in the DGHPSOGS algorithm, being the worst one in terms of computational time. The ARPSO algorithm outperformed GPSO, DGHPSOGS and SAPSO algorithms on dimensions 20 and 30. However, ARPSO is one of the worst algorithm when applied to problems to find the global minimum, as previously discussed in Table 5.

Table 8 – Average CPU time (in seconds) for each number of dimensions.

d	**ARPSO**	**GPSO**	**DGHPSOGS**	**SAPSO**
2	0.2957	3.4728	1.6537	**0.1787**
10	4.9236	5.64734	10.1958	**3.698**
20	**7.2145**	12.7473	31.9430	15.4181
30	**8.2373**	18.6776	54.838	31.7664

Source: produced by the author.

Other information extracted from Table 7 is the one presented in Table 9, which shows that the SAPSO algorithm outperformed the other PSOs regarding the average number of iterations required to converge to a feasible solution. The ARPSO algorithm is better than GPSO and DGHPSOGS on 2 dimensional problems. The DGHPSOGS algorithm outperformed GPSO and ARPSO algorithms on 10 dimensional problems, while GPSO algorithm is better than DGHPSOGS and ARPSO algorithms on 20 and 30 dimensional problems.

Table 9 – Average number of iterations required to converge to a feasible solution.

d	**ARPSO**	**GPSO**	**DGHPSOGS**	**SAPSO**
2	111.075	1895.575	452.825	**49.975**
10	2014.3375	1364.5625	1181.9187	**439.825**
20	2897.7875	1931.7437	2420.3875	**1185.825**
30	3274.1125	2152.5937	3080.3437	**1784.85**

Source: produced by the author.

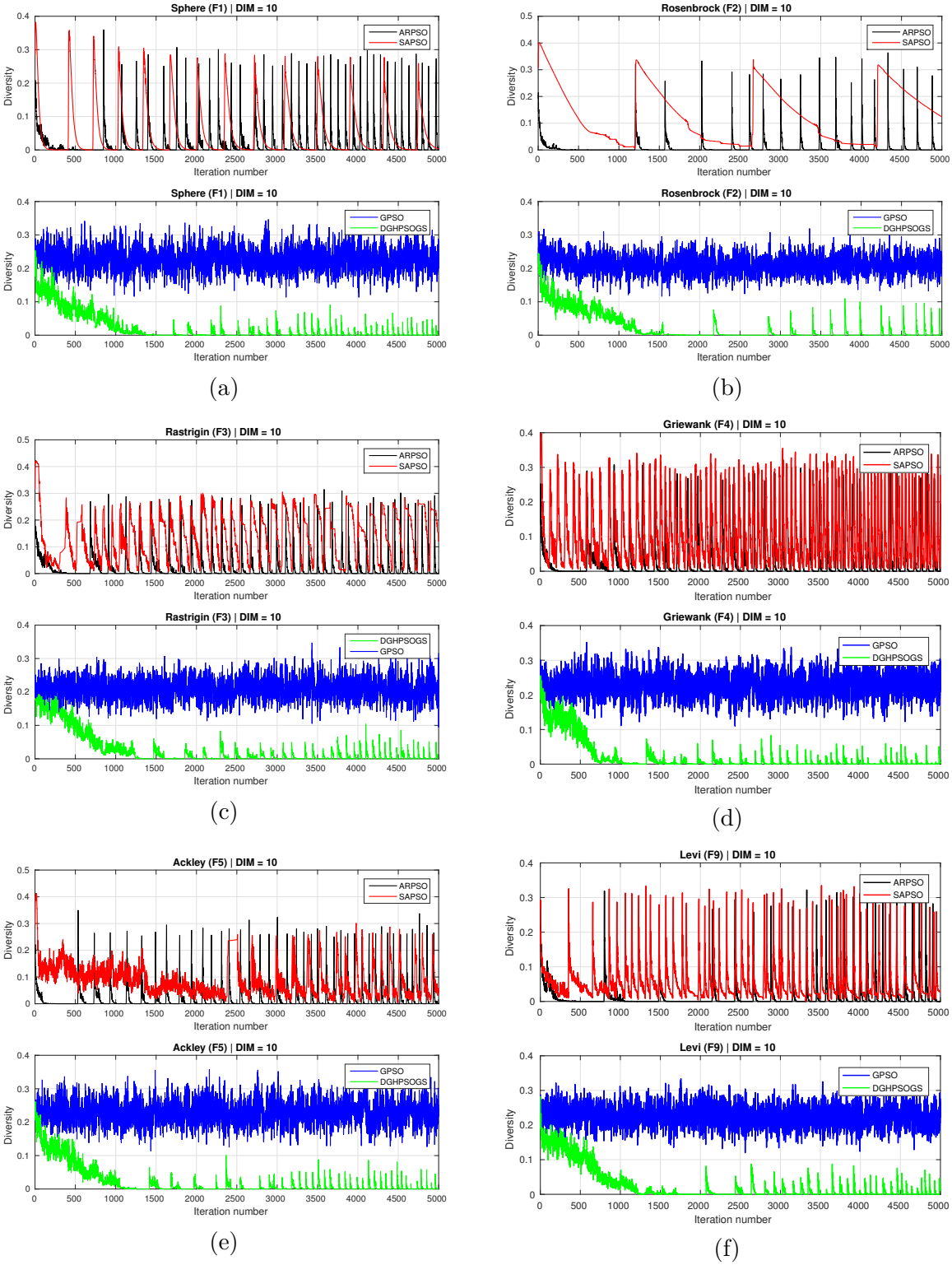
As can be seen from the above experiments, the SAPSO algorithm guarantees the global minimum (or at least near the global minimum) in all test functions. The gradient-based information is crucial and assists the algorithm in the search process, although it takes time to calculate the partial derivatives in higher dimensional problems. However, the total number of iterations required to find the global minimum was significantly decreased, which might be an indicative that the semi-autonomous particles are investigating the search space in a parsimonious fashion.

4.1.5 Diversity control analysis

The following experiment is conducted on the test functions with ten dimensions (without loss of generality for higher dimensions). The attractive and repulsive schemes, as represented by SAPSO, ARPSO, and DGHPSOGS, are shown in the Figure 17. This figure shows the diversity values of the swarm in the PSOs for six test functions. Each repulsion phase is a chance of escaping from a local minimum, as the test functions are highly multimodal with many local minima through the search space, this becomes an important feature incorporated in the proposed algorithm.

The SAPSO and ARPSO algorithms keep the diversity of the swarm adaptively in the whole search process. However, ARPSO loses its diversity faster than SAPSO after a repulsion phase. This effect is stronger in Figures 17b and 17e, but it is also noted in all test functions. It is worth noting that after a repulsion phase, SAPSO algorithm applies the state {10} (Figure 8), which means the particles are in the attraction phase and they are autonomous to evaluate their own area in the search space through negative gradient direction. Note that, this state leads to a slow decreasing of diversity value along

Figure 17 – The diversity values of four PSOs on five test functions: (a) Sphere; (b) Rosenbrock; (c) Rastrigin; (d) Griewank; (e) Ackley; (f) Levi.



Source: produced by the author.

the iterations and brings a stronger capability of exploiting different areas of the search space simultaneously and properly, which in turns also minimizes the number of iterations to investigate a valley and reduces the effect of random walk (due to the deterministic directions of gradients).

The DGHPSOGS algorithm applies the repulsion phase for a short period of time, thus the particle is not far enough apart to jump out from one local minimum to another when compared to the proposed SAPSO algorithm. Note that, the DGHPSOGS algorithm defines the lower bound only (review Section 2.9.3), i.e., attraction and repulsion phases are controlled by only one threshold. This mechanism leads the algorithm to a faster switching between phases of attraction and repulsion during the search process. This behavior can be seen in all test functions. On the other hand, the GPSO algorithm has no diversity control at all. One can note that there is no decrease of the diversity value along the iterations in the GPSO algorithm in all test functions. The GPSO algorithm is a pure classical PSO running under the hood with a quasi Newton-Raphson method applied over the global best position only. In other words, each particle has the contradictory and probabilistic decision to follow between the cognitive and social directions at the same time in each iteration.

This model of algorithm is excellent in unimodal functions such as Sphere and Rosenbrock, due to the deterministic method applied at each iteration. However, for multimodal functions, a prominent result depends of a probabilistic combination of steps for at least one particle be in the correct valley, e.g., a valley where the global minimum is, and to properly exploit this region in such manner that the global best position turns into its own position. In this particular case, the GPSO algorithm will take that global best position and lead it to a fast convergence to the minimum and the particles of the swarm will not have enough iterations to converge to a cluster.

The experiment presented in Table 10 relates the average of number of times a repulsion phase occurs and the average number of iterations required to achieve the global minimum is provided. The experiment confirms that SAPSO algorithm applies less repulsion phases and still find the global minimum with less iterations. Besides, no PSO algorithms found the exact global minimum in Rosenbrock and Levi functions. Thus, the maximum number of iterations (5000) was used as a stop criterion. The SAPSO algorithm used the stop criterion in Levi N.13 function, but less repulsion phases were applied. However, the SAPSO algorithm still found better results with less repulsion phases (Table 5) on Rosenbrock and Levi N.13 functions. This fact indicates an appropriate trade-off of exploration and exploitation and suggests a better use of the attraction and repulsion schemes.

The ARPSO algorithm always required more iterations to reach the global minimum, followed by the DGHPSOGS algorithm on all test function, except for the Levi N.13

Table 10 – Average number of repulsions and stop criteria match by three PSOs on ten test functions.

Function	**ARPSO**		**DGHPSOGS**		**SAPSO**	
	repulsion	iteration	repulsion	iteration	repulsion	iteration
Sphere	1.3	930.15	**0**	**121.5**	3	776.8
Rosenbrock	14.05	5000	12.45	5000	**3**	5000
Rastrigin	2.95	1413.7	1	499.8	**0.55**	**179.05**
Griewank	1.5	854.6	**0**	85.65	0.05	**41.25**
Ackley	1.75	911.85	**0**	83.95	**0**	**28.85**
Ellipsoid	4.95	1355.8	3.15	661.25	**2.55**	**174.25**
Schaffer N.2	**0**	181.55	**0**	13.45	0.1	**12.9**
Alpine	4.9500	1566.1	3.2500	766.7	**2.8**	**19.75**
Levi	**25.35**	5000	30.05	5000	63.15	5000
Levi N.13	0.65	**392.45**	0.15	1403.5	**0.1**	5000

Source: produced by the author.

function. Despite this fact, the number of repulsion phases required by the DGHPSOGS algorithm was smaller than the one required by ARPSO algorithm in all cases (except on the Levi function). Figure 17 brought this idea by presenting a slow decrease of the diversity value of the DGHPSOGS algorithm along the iterations. For instance, this behavior contributes to better exploit the search space of the Sphere function, since the gradient information is also used by the DGHPSOGS algorithm.

4.2 Results: rotation and information exchange

This section outlines the methodology of the experiments following the rules stated by Clerc (Section 2.10). The test functions described in this work are based on CEC 2017 benchmark problems in which single objective real-parameter numerical optimization is under consideration. All optimization functions are randomly shifted and rotated. As there are seldom any relation among variables in real-world optimization problems, the rotate matrices are divided into subcomponents randomly and each subcomponents are generated from standard normally distributed entries by Gram-Schmidt ortho-normalization (CHENEY; KINCAID, 2008).

In the above context, the author cover Clerc's rule 1, as the global optimum of any test function is not in the center, axis or diagonal of the problem space. Rule 2 is devoted to the estimated performance whose convergence is reliable. In Section 3.2.3, an empirical test to define a reasonable value for E (number of experiments) was described. Rule 3a and 3b are covered by two performance criterion: error and success rates. The computational time is also carried out for each PSO version. Additionally, two non-parametric statistical

hypothesis tests are performed to strengthen the analyses of results. Rule 4 and 5 were previously discussed in Section 2.10.

As far as the author know, no other author has investigated this topic. Herein, this thesis evaluates the influence of the late and fast exchange of information among particles in both rotationally variant and invariant versions of PSO, named according to Table 11.

Table 11 – Notation of each algorithm under evaluation. The under-script stands for the rotational version, while the superscript indicates the type of information exchange.

Algorithm	**Notation**	**Description**
1	Ψ_{var}^{late}	Rotational **variant** and **late** information exchange
2	Ψ_{var}^{fast}	Rotational **variant** and **fast** information exchange
3	Ψ_{inv}^{late}	Rotational **invariant** and **late** information exchange
4	Ψ_{inv}^{fast}	Rotational **invariant** and **fast** information exchange

Source: produced by the author.

In order to make the results as reliable as possible, important rules pointed by Maurice Clerc (CLERC, 2012a) will be followed up, which implies among other complex test functions, statistical tests and law of large numbers. The following sections provide many details about the experiments of this work: Section 4.2.1 presents the optimization problems and their characteristics; Section 4.2.2 reports the simulation results found by the experimental tests; Section 4.2.3 describes and shows the statistical hypothesis tests.

4.2.1 CEC 2017 benchmark problems

The next experimental tests are based on ten optimization problems, including shifted, rotated and composition test functions. Before describe them, Table 12 defines the basic form of all test functions. As the composition functions are formed by more than one function, the total of basic functions is thirteen.

Each test function offers different challenges for any optimization algorithm, such as: many local minima surrounding the global minimum, nearly flat regions, d-dimensional continuous space for searching, and decentralized global minimum. The functions are used to assert the qualities of each PSO version under evaluation, particularly on multimodal functions where the algorithms must be able to avoid local minima and fine-tune the global optimum. As the functions are shifted, rotated, and composed, Table 13 describes

Table 12 – Basic test functions from CEC 2017 benchmark suite.

Function	**Equation**	Mode
Bent Cigar (b_1)	$x_1^2 + 10^6 \sum_{i=2}^{d} x_i^2$	unimodal
Zakharov (b_2)	$\sum_{i=1}^{d} x_i^2 + \left(\sum_{i=1}^{d} 0.5x_i\right)^2 + \left(\sum_{i=1}^{d} 0.5x_i\right)^4$	unimodal
Rosenbrock (b_3)	$\sum_{i=1}^{d-1} 100\left(x_i^2 - x_{i+1}\right)^2 + \left(x_i - 1\right)^2$	multimodal
Rastrigin (b_4)	$\sum_{i=1}^{d} \left(x_i^2 - 10\cos\left(2\pi x_i\right) + 10\right)$	multimodal
Levy (b_5)	$\sin^2\left(\pi z_1\right) + \sum_{i=1}^{d-1}\left(z_i - 1\right)^2\left(1 + 10\sin^2\left(\pi z_i + 1\right)\right) + \left(z_d - 1\right)^2\left(1 + \sin^2\left(2\pi z_d\right)\right)$ where $z_i = 1 + \frac{x_i - 1}{4}, \forall\, i \in \{1, 2, \ldots, d\}$	multimodal
Modified Schwefel (b_6)	$418.9829d - \sum_{i=1}^{d} g(z_i)$ where $z_i = x_i + 4.2096874622750361\text{E+}002$, $g(z_i) \begin{cases} z_i \sin\left(\lvert z_i\rvert^{1/2}\right) & \text{if } \lvert z_i\rvert \leq 500 \\ \left(500 - \bmod\left(z_i, 500\right)\right)\sin\left(\sqrt{\lvert 500 - \bmod\left(z_i, 500\right)\rvert}\right) - \frac{(z_i - 500)^2}{10000d} & \text{if } z_i > 500 \\ \left(\bmod\left(\lvert z_i\rvert, 500\right) - 500\right)\sin\left(\sqrt{\lvert \bmod\left(\lvert z_i\rvert, 500\right) - 500\rvert}\right) - \frac{(z_i + 500)^2}{10000d} & \text{if } z_i < -500 \end{cases}$	multimodal
Happycat (b_7)	$\left\lvert \sum_{i=1}^{d} x_i^2 - d\right\rvert^{1/4} + \left(0.5\sum_{i=1}^{d} x_i^2 + \sum_{i=1}^{d} x_i\right)/d + 0.5$	multimodal
Ackley (b_8)	$-20\exp\left(-0.2\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}\right) - \exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos\left(2\pi x_i\right)\right) + 20 + \exp\left(1\right)$	multimodal
Discus (b_9)	$10^6 x_1^2 + \sum_{i=2}^{d} x_i^2$	unimodal
Expanded Schaffer's F6 (b_{10})	$\sum_{i=1}^{d-1} g(x_i, x_{i+1}) + g(x_d, x_1)$ $g(x, y) = 0.5 + \frac{\left(\sin^2\left(\sqrt{x^2 + y^2}\right) - 0.5\right)}{\left(1 + 0.001\left(x^2 + y^2\right)\right)^2}$	multimodal
Griewank (b_{11})	$1 + \frac{1}{4000}\sum_{i=1}^{d} x_i^2 - \prod_{i=1}^{d}\cos\left(\frac{x_i}{\sqrt{i}}\right)$	multimodal
HGBat (b_{12})	$\left\lvert \left(\sum_{i=1}^{d} x_i^2\right)^2 - \left(\sum_{i=1}^{d} x_i\right)^2\right\rvert^{1/2} + \left(0.5\sum_{i=1}^{d} x_i^2 + \sum_{i=1}^{d} x_i\right)/d + 0.5$	unimodal
High Conditioned Elliptic (b_{13})	$\sum_{i=1}^{d} (10^6)^{\frac{i-1}{d-1}} x_i^2$	multimodal

Source: produced by the author.

the transformations required to achieve such modifications with their respectively global minimum values defined as F_k^*.

Note that, the basic functions used to form the composition functions $f_{k\in\{7,\ldots,10\}}$ are shifted and rotated. The following points describe the meaning of variables, matrices, and auxiliary functions:

- $f_{k\in\{7,\ldots,10\}}$: composition functions;
- $\mathbf{M}_{i\in\{1,\ldots,13\}}$: rotation matrix. Different rotation matrix are used for each basic function;
- $\vec{\mathbf{o}}_{i\in\{1,\ldots,13\}}$: shifted optimum position in which $o_{ij} \sim \mathcal{U}(-80, 80)$;
- $-100 \leq x_{j\in\{1,\ldots,d\}} \leq 100$: search range for all test functions;
- N: number of basic functions;
- $g_{i\in\{1,\ldots,N\}}$: ith basic function used to construct the composition function;
- $bias_{i\in\{1,\ldots,N\}}$: defines which optimum is the global optimum;

Table 13 – Shifted, rotated, and composition test function.

Function	**Equation**	$F_k^* = f_k(\vec{\mathbf{x}}_{opt})$
Bent Cigar (f_1)	$b_1(\mathbf{M}_1(\vec{\mathbf{x}} - \vec{\mathbf{o}}_1)) + F_1^*$	100
Zakharov (f_2)	$b_2(\mathbf{M}_2(\vec{\mathbf{x}} - \vec{\mathbf{o}}_2)) + F_2^*$	300
Rosenbrock (f_3)	$b_3(\mathbf{M}_3(\frac{2.048(\vec{\mathbf{x}}-\vec{\mathbf{o}}_3)}{100}) + 1) + F_3^*$	400
Rastrigin (f_4)	$b_4(\mathbf{M}_4(\vec{\mathbf{x}} - \vec{\mathbf{o}}_4)) + F_4^*$	500
Levy (f_5)	$b_5(\mathbf{M}_5(\frac{5.12(\vec{\mathbf{x}}-\vec{\mathbf{o}}_5)}{100})) + F_5^*$	900
Schwefel (f_6)	$b_6(\mathbf{M}_6(\frac{1000(\vec{\mathbf{x}}-\vec{\mathbf{o}}_6)}{100})) + F_6^*$	1000
Composition (f_7)	$\sum_{i=1}^{N} \omega_i[\lambda_i g_i(\vec{\mathbf{x}}) + bias_i] + F_7^*$ where $N = 5$, $\sigma = [10, 20, 30, 40, 50]$, $\lambda = [10, 1, 10, 1\text{E}{-6}, 1]$, $bias = [0, 100, 200, 300, 400]$, $g = [b_4, b_7, b_8, b_9, b_3]$	2500
Composition (f_8)	$\sum_{i=1}^{N} \omega_i[\lambda_i g_i(\vec{\mathbf{x}}) + bias_i] + F_8^*$ where $N = 5$, $\sigma = [10, 20, 20, 30, 40]$, $\lambda = [1\text{E}{-26}, 10, 1\text{E}{-6}, 10, 5\text{E}{-4}]$, $bias = [0, 100, 200, 300, 400]$, $g = [b_{10}, b_6, b_{11}, b_3, b_4]$	2600
Composition (f_9)	$\sum_{i=1}^{N} \omega_i[\lambda_i g_i(\vec{\mathbf{x}}) + bias_i] + F_9^*$ where $N = 6$, $\sigma = [10, 20, 30, 40, 50, 60]$, $\lambda = [10, 10, 2.5, 1\text{E}{-26}, 1\text{E}{-6}, 5\text{E}{-4}]$, $bias = [0, 100, 200, 300, 400, 500]$, $g = [b_{12}, b_4, b_6, b_1, b_{13}, b_{10}]$	2700
Composition (f_{10})	$\sum_{i=1}^{N} \omega_i[\lambda_i g_i(\vec{\mathbf{x}}) + bias_i] + F_{10}^*$ where $N = 6$, $\sigma = [10, 20, 30, 40, 50, 60]$, $\lambda = [10, 10, 1\text{E}{-6}, 1, 1, 5\text{E}{-4}]$, $bias = [0, 100, 200, 300, 400, 500]$, $g = [b_8, b_{11}, b_9, b_3, b_7, b_{10}]$	2800

Source: produced by the author.

- $\sigma_{i\in\{1,\ldots,N\}}$: controls each g_i's coverage range;
- $\lambda_{i\in\{1,\ldots,N\}}$: controls each g_i's height;

- $w_{i\in\{1,\ldots,N\}}$: weight value for each g_i defined as:

$$w_i = \frac{1}{\sqrt{\sum_{j=1}^{d}(x_j - o_{ij})^2}} \exp\left(-\frac{\sum_{j=1}^{d}(x_j - o_{ij})^2}{2d\sigma_i^2}\right). \tag{4.1}$$

- $\omega_{i\in\{1,\ldots,N\}}$ is the normalized version of w_i defined as:

$$\omega_i = \frac{w_i}{\sum_{i=1}^{N} w_i}. \tag{4.2}$$

In the composition functions, when $\vec{\mathbf{x}} = \vec{\mathbf{o}}_i$, then:

$$\omega_j = \begin{cases} 1 & j = i \\ 0 & j \neq i \end{cases} \quad \forall\, i, j \in \{1, 2, \ldots, N\}. \tag{4.3}$$

An adjustment in the form of $F_i = F_i - F_i^*$ is applied on each basic function represented by g_i to produce global optimum values equal to zero. Finally, the global optimum value of the composition function is defined as the smallest bias value according to

$$f_{k\in\{7,\ldots,10\}}(\vec{\mathbf{x}}_{opt}) = bias_i + F_k^*. \tag{4.4}$$

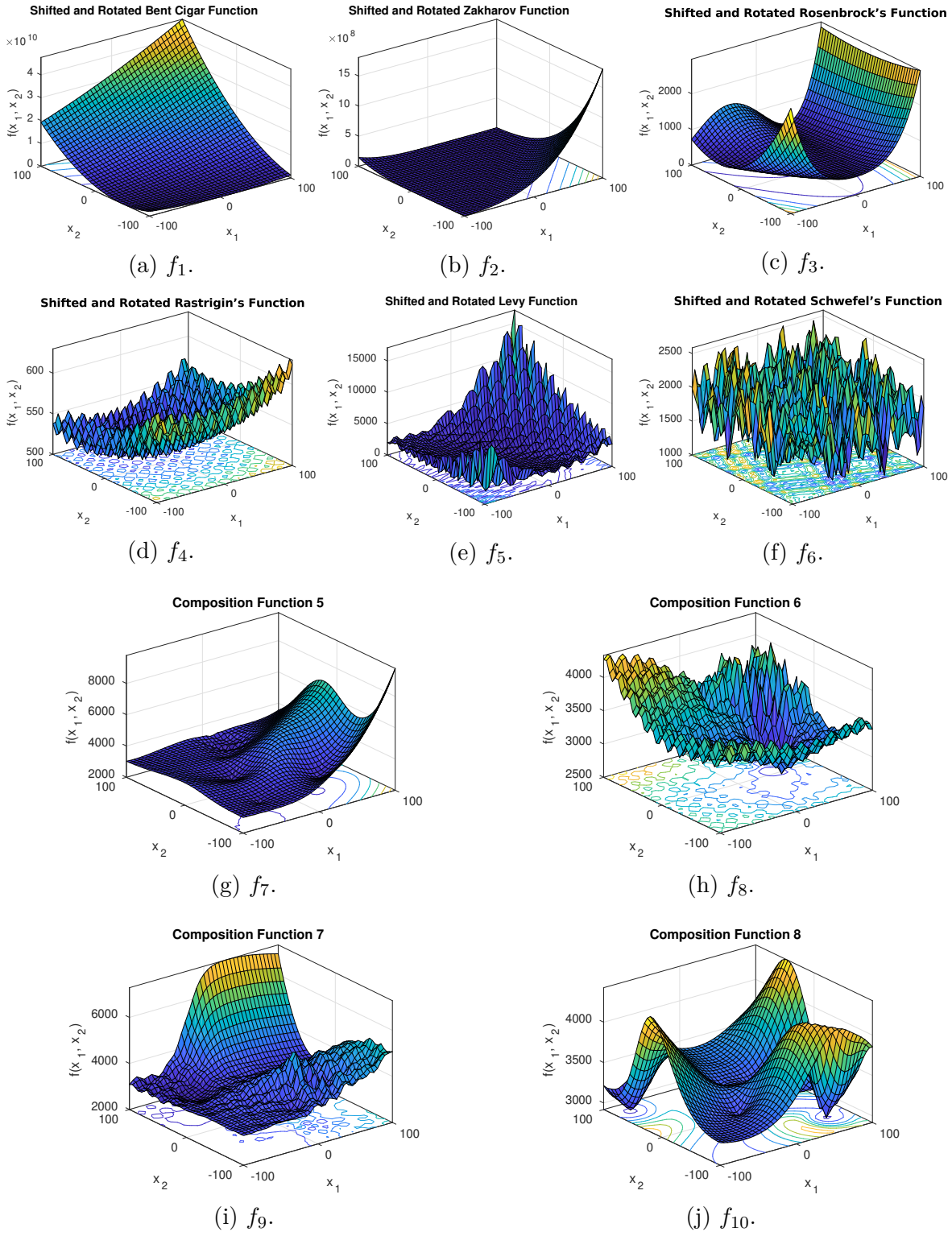
A bi-dimensional representation of each function is depicted in Figure 18. In this thesis, the experiments are conducted on functions defined with 2, 10, 30, and 50 dimensions.

4.2.2 Toward the minimum error rate

This section describes and presents the simulations conducted by this experiments. The parameter settings of the algorithms are inspired by the works (EBERHART; KENNEDY, 1995; KENNEDY; EBERHART, 1995; SHI; EBERHART, 1998a) and summarized in Table 14. The cognitive and social coefficients are both set as 2, which provide an acceptable trade-off between exploration and exploitation, as the expected value of random factors (either ϕ or $\vec{\phi}$) is 1 in all dimensions, providing the same influence of the coefficients over the particles. The chosen bound handling mechanism for all PSO versions is *Nearest*.

The inertia weight is ruled by a linear decreasing function over the generations varying from 0.9 to 0.4 in which is based on (SHI; EBERHART, 1999; RATNAWEERA; HALGAMUGE; WATSON, 2004). The maximum number of iterations is empirically set as 5000 for each execution. The number of execution is 134, as previously discovered in Section 3.2.3. Note that, each particle evaluates the test function only one time per

Figure 18 – Bi-dimensional surface of shifted, rotated and composition test functions of CEC 2017 benchmark suite.



(a) f_1. (b) f_2. (c) f_3.

(d) f_4. (e) f_5. (f) f_6.

(g) f_7. (h) f_8.

(i) f_9. (j) f_{10}.

Source: produced by the author.

iteration, as a mandatory step right after the particle movement. Thus, the maximum number of function evaluations is 100,000 (the conversion from generation to function

Table 14 – Parameter settings for all versions of PSO.

Variable	**Value**
Number of particles (n)	20
Number of dimensions (d)	[2, 10, 30, 50]
Cognitive coefficient (c_1)	2
Social coefficient (c_2)	2
Inertia weight (w)	$[.9 \rightarrow .4]$
Number of Iterations (T)	5000
Number of executions (E)	134

Source: produced by the author.

evaluations is straightforward: $n \times T = 20 \times 5000$).

The PSO algorithms stop in two conditions: 1) the error rate is below a predefined threshold ξ, which is estimated by the difference of the function values between global memory position and the known global minimum position, i.e., $f(\vec{\mathbf{x}}) - f(\vec{\mathbf{x}}_{opt}) \leq \xi$; otherwise the algorithms stop when 2) the maximum number of iterations is reached.

Throughout this section, a collection of performance criteria is used to evaluate the four versions of PSO. Since 134 executions are performed by each test function and in each dimensional problem, the results are averaged over all executions. All experiments are conducted regarding two error thresholds. Thus, a feasible result is considered if the error is lower than $\xi = 10^{-2}$ (first experiment) or $\xi = 10^{-8}$ (second experiment). Table 15 exhibits the average error rate for all test functions. The best results are in bold and the last row corresponds to a simple rank based on the winner count.

The test functions are difficult to optimize due to the shifted and rotated features. The versions of PSO are very basic, without sophisticated mechanism to escape from local optima or to properly investigate promising areas of the search space when necessary. Therefore, one can expect that the lack of elaborated features might influence the results. It is worth noting that complex functions are of interest, as the main goal of this work is to distinguish the performance of PSO algorithms to some extent.

Another important information from Table 15 is the naive score for best results (last row). This table announces the phenomenon previously discussed by Rule 3a in which by considering late information exchange only, no such a winner is observed: for $\xi = 10^{-2}$, Ψ_{inv}^{late} won, but when $\xi = 10^{-8}$, Ψ_{var}^{late} comes in first. The same phenomenon occurred when fast exchange information takes place, which in turns recalls the "no free lunch theorem". The phenomenon is also observed in this experiment, as one can clearly verify that there is no such PSO algorithm with the best performance among all test functions. Even Ψ_{var}^{fast}, which is the overall winner considering the rank, is not the best algorithm in all optimization problems. Later in Section 4.2.3, two statistical hypothesis

Table 15 – Average error rate obtained in the benchmark functions.

f_i	d	$\xi = 10^{-2}$				$\xi = 10^{-8}$			
		Ψ_{var}^{late}	Ψ_{inv}^{late}	Ψ_{var}^{fast}	Ψ_{inv}^{fast}	Ψ_{var}^{late}	Ψ_{inv}^{late}	Ψ_{var}^{fast}	Ψ_{inv}^{fast}
f_1	2	$\mathbf{1.0363 \cdot 10^{3}}$	$1.2301 \cdot 10^{3}$	$1.8421 \cdot 10^{3}$	$1.9066 \cdot 10^{3}$	$\mathbf{1.3004 \cdot 10^{3}}$	$1.4360 \cdot 10^{3}$	$1.6006 \cdot 10^{3}$	$1.7355 \cdot 10^{3}$
	10	$9.9705 \cdot 10^{8}$	$9.0757 \cdot 10^{8}$	$9.1413 \cdot 10^{8}$	$\mathbf{6.7557 \cdot 10^{8}}$	$9.3920 \cdot 10^{8}$	$1.0377 \cdot 10^{9}$	$\mathbf{8.2309 \cdot 10^{8}}$	$9.1577 \cdot 10^{8}$
	30	$1.3576 \cdot 10^{10}$	$1.2518 \cdot 10^{10}$	$\mathbf{8.7001 \cdot 10^{9}}$	$9.5002 \cdot 10^{9}$	$1.4714 \cdot 10^{10}$	$1.3262 \cdot 10^{10}$	$9.6544 \cdot 10^{9}$	$\mathbf{9.2122 \cdot 10^{9}}$
	50	$4.5745 \cdot 10^{10}$	$4.5875 \cdot 10^{10}$	$\mathbf{3.4734 \cdot 10^{10}}$	$3.4994 \cdot 10^{10}$	$4.4318 \cdot 10^{10}$	$4.5726 \cdot 10^{10}$	$\mathbf{3.2378 \cdot 10^{10}}$	$3.2439 \cdot 10^{10}$
f_2	2	$5.1064 \cdot 10^{-3}$	$5.3003 \cdot 10^{-3}$	$5.3628 \cdot 10^{-3}$	$\mathbf{4.8076 \cdot 10^{-3}}$	$5.3254 \cdot 10^{-9}$	$5.6937 \cdot 10^{-9}$	$\mathbf{5.1425 \cdot 10^{-9}}$	$2.7636 \cdot 10$
	10	$\mathbf{2.4761 \cdot 10^{3}}$	$4.4130 \cdot 10^{3}$	$3.9344 \cdot 10^{3}$	$3.4471 \cdot 10^{3}$	$5.4495 \cdot 10^{3}$	$4.1209 \cdot 10^{3}$	$\mathbf{3.6723 \cdot 10^{3}}$	$4.0285 \cdot 10^{3}$
	30	$8.0586 \cdot 10^{4}$	$8.3407 \cdot 10^{4}$	$\mathbf{7.9083 \cdot 10^{4}}$	$8.1820 \cdot 10^{4}$	$7.9541 \cdot 10^{4}$	$8.8074 \cdot 10^{4}$	$\mathbf{7.7209 \cdot 10^{4}}$	$8.5724 \cdot 10^{4}$
	50	$\mathbf{2.3550 \cdot 10^{5}}$	$2.4895 \cdot 10^{5}$	$2.4622 \cdot 10^{5}$	$2.3953 \cdot 10^{5}$	$2.5081 \cdot 10^{5}$	$2.4484 \cdot 10^{5}$	$\mathbf{2.2991 \cdot 10^{5}}$	$2.4180 \cdot 10^{5}$
f_3	2	$5.5015 \cdot 10^{-3}$	$5.0065 \cdot 10^{-3}$	$5.1903 \cdot 10^{-3}$	$\mathbf{4.8448 \cdot 10^{-3}}$	$5.7551 \cdot 10^{-9}$	$5.9471 \cdot 10^{-9}$	$5.3904 \cdot 10^{-9}$	$\mathbf{5.2361 \cdot 10^{-9}}$
	10	$5.8704 \cdot 10$	$\mathbf{4.4042 \cdot 10}$	$4.5266 \cdot 10$	$4.6580 \cdot 10$	$5.9293 \cdot 10$	$5.6364 \cdot 10$	$\mathbf{4.4452 \cdot 10}$	$5.5527 \cdot 10$
	30	$1.7617 \cdot 10^{3}$	$1.7809 \cdot 10^{3}$	$\mathbf{1.2648 \cdot 10^{3}}$	$1.3462 \cdot 10^{3}$	$1.6366 \cdot 10^{3}$	$1.7029 \cdot 10^{3}$	$1.2119 \cdot 10^{3}$	$\mathbf{1.1979 \cdot 10^{3}}$
	50	$6.2898 \cdot 10^{3}$	$6.2036 \cdot 10^{3}$	$\mathbf{4.6742 \cdot 10^{3}}$	$5.1340 \cdot 10^{3}$	$6.3718 \cdot 10^{3}$	$6.9435 \cdot 10^{3}$	$\mathbf{4.3288 \cdot 10^{3}}$	$4.8315 \cdot 10^{3}$
f_4	2	$5.3972 \cdot 10^{-3}$	$4.9741 \cdot 10^{-3}$	$\mathbf{4.7022 \cdot 10^{-3}}$	$4.7392 \cdot 10^{-3}$	$4.9369 \cdot 10^{-9}$	$5.1772 \cdot 10^{-9}$	$\mathbf{4.7552 \cdot 10^{-9}}$	$5.2896 \cdot 10^{-9}$
	10	$\mathbf{2.3572 \cdot 10}$	$2.5527 \cdot 10$	$2.3868 \cdot 10$	$2.5115 \cdot 10$	$2.3358 \cdot 10$	$2.4424 \cdot 10$	$\mathbf{2.2510 \cdot 10}$	$2.4346 \cdot 10$
	30	$1.6178 \cdot 10^{2}$	$1.6078 \cdot 10^{2}$	$1.4807 \cdot 10^{2}$	$\mathbf{1.4639 \cdot 10^{2}}$	$1.5411 \cdot 10^{2}$	$1.5730 \cdot 10^{2}$	$\mathbf{1.4007 \cdot 10^{2}}$	$1.4494 \cdot 10^{2}$
	50	$3.5366 \cdot 10^{2}$	$3.5093 \cdot 10^{2}$	$3.2735 \cdot 10^{2}$	$\mathbf{3.1740 \cdot 10^{2}}$	$3.4570 \cdot 10^{2}$	$3.4570 \cdot 10^{2}$	$\mathbf{3.2113 \cdot 10^{2}}$	$3.2357 \cdot 10^{2}$
f_5	2	$4.9470 \cdot 10^{-3}$	$\mathbf{4.7122 \cdot 10^{-3}}$	$4.8215 \cdot 10^{-3}$	$4.7864 \cdot 10^{-3}$	$5.4638 \cdot 10^{-9}$	$5.5404 \cdot 10^{-9}$	$\mathbf{5.3639 \cdot 10^{-9}}$	$5.7088 \cdot 10^{-9}$
	10	8.0214	1.9830 · 10	4.8735	**3.4676**	$3.2687 \cdot 10$	$2.0876 \cdot 10$	4.7420	**4.5227**
	30	$3.6308 \cdot 10^{3}$	$3.6055 \cdot 10^{3}$	$3.5140 \cdot 10^{3}$	$\mathbf{2.9748 \cdot 10^{3}}$	$3.7778 \cdot 10^{3}$	$3.8003 \cdot 10^{3}$	$\mathbf{3.2551 \cdot 10^{3}}$	$3.2827 \cdot 10^{3}$
	50	$1.6757 \cdot 10^{4}$	$1.7278 \cdot 10^{4}$	$\mathbf{1.4677 \cdot 10^{4}}$	$1.5941 \cdot 10^{4}$	$1.7939 \cdot 10^{4}$	$1.6342 \cdot 10^{4}$	$\mathbf{1.5444 \cdot 10^{4}}$	$1.6145 \cdot 10^{4}$
f_6	2	**3.4745**	7.1749	7.0379	6.5470	6.2300	4.3138	**3.8801**	7.9505
	10	$\mathbf{6.3182 \cdot 10^{2}}$	$6.5214 \cdot 10^{2}$	$6.7792 \cdot 10^{2}$	$6.3601 \cdot 10^{2}$	$6.7361 \cdot 10^{2}$	$6.3172 \cdot 10^{2}$	$\mathbf{6.1219 \cdot 10^{2}}$	$6.5672 \cdot 10^{2}$
	30	$4.0033 \cdot 10^{3}$	$4.0564 \cdot 10^{3}$	$4.0807 \cdot 10^{3}$	$\mathbf{3.9996 \cdot 10^{3}}$	$\mathbf{4.0307 \cdot 10^{3}}$	$4.1664 \cdot 10^{3}$	$4.1189 \cdot 10^{3}$	$4.0385 \cdot 10^{3}$
	50	$7.9329 \cdot 10^{3}$	$7.7368 \cdot 10^{3}$	$7.8001 \cdot 10^{3}$	$\mathbf{7.6545 \cdot 10^{3}}$	$7.7064 \cdot 10^{3}$	$7.6689 \cdot 10^{3}$	$7.7400 \cdot 10^{3}$	$\mathbf{7.5421 \cdot 10^{3}}$
f_7	2	$1.5002 \cdot 10$	$1.2829 \cdot 10$	**9.7081**	$1.5364 \cdot 10$	$1.9088 \cdot 10$	$1.3433 \cdot 10$	$\mathbf{1.3433 \cdot 10}$	$1.4179 \cdot 10$
	10	$4.6514 \cdot 10^{2}$	$4.5811 \cdot 10^{2}$	$\mathbf{4.5345 \cdot 10^{2}}$	$4.5535 \cdot 10^{2}$	$4.6972 \cdot 10^{2}$	$4.6074 \cdot 10^{2}$	$\mathbf{4.5362 \cdot 10^{2}}$	$4.5366 \cdot 10^{2}$
	30	$7.6633 \cdot 10^{2}$	$7.4949 \cdot 10^{2}$	$\mathbf{6.1469 \cdot 10^{2}}$	$6.2024 \cdot 10^{2}$	$7.5143 \cdot 10^{2}$	$7.9727 \cdot 10^{2}$	$6.0544 \cdot 10^{2}$	$\mathbf{6.0299 \cdot 10^{2}}$
	50	$3.3030 \cdot 10^{3}$	$3.3085 \cdot 10^{3}$	$\mathbf{2.1421 \cdot 10^{3}}$	$2.4317 \cdot 10^{3}$	$3.5435 \cdot 10^{3}$	$3.1147 \cdot 10^{3}$	$2.2097 \cdot 10^{3}$	$\mathbf{2.1182 \cdot 10^{3}}$
f_8	2	$6.7228 \cdot 10^{-3}$	$6.8841 \cdot 10^{-3}$	$6.9635 \cdot 10^{-3}$	$\mathbf{6.6855 \cdot 10^{-3}}$	$\mathbf{6.8283 \cdot 10^{-9}}$	$6.9065 \cdot 10^{-9}$	$7.1369 \cdot 10^{-9}$	$6.8762 \cdot 10^{-9}$
	10	$6.9794 \cdot 10^{2}$	$6.3698 \cdot 10^{2}$	$6.7221 \cdot 10^{2}$	$\mathbf{5.7071 \cdot 10^{2}}$	$6.9387 \cdot 10^{2}$	$6.2725 \cdot 10^{2}$	$\mathbf{6.0537 \cdot 10^{2}}$	$6.2433\text{e} \cdot 10^{2}$
	30	$4.1126 \cdot 10^{3}$	$4.2228 \cdot 10^{3}$	$3.8694 \cdot 10^{3}$	$\mathbf{3.7022 \cdot 10^{3}}$	$4.1369 \cdot 10^{3}$	$4.1482 \cdot 10^{3}$	$3.9654 \cdot 10^{3}$	$\mathbf{3.8485 \cdot 10^{3}}$
	50	$9.4762 \cdot 10^{3}$	$9.0946 \cdot 10^{3}$	$\mathbf{8.2951 \cdot 10^{3}}$	$8.4071 \cdot 10^{3}$	$8.9468 \cdot 10^{3}$	$9.7969 \cdot 10^{3}$	$8.6902 \cdot 10^{3}$	$\mathbf{8.3908 \cdot 10^{3}}$
f_9	2	$\mathbf{7.6738 \cdot 10^{-3}}$	$7.8089 \cdot 10^{-3}$	$8.1611 \cdot 10^{-3}$	$7.7787 \cdot 10^{-3}$	$4.6515 \cdot 10^{-4}$	$4.8005 \cdot 10^{-4}$	$\mathbf{4.3372 \cdot 10^{-4}}$	$4.3513 \cdot 10^{-4}$
	10	$4.2670 \cdot 10^{2}$	$\mathbf{4.2401 \cdot 10^{2}}$	$4.2950 \cdot 10^{2}$	$4.3589 \cdot 10^{2}$	$4.2895 \cdot 10^{2}$	$4.3265 \cdot 10^{2}$	$4.2778 \cdot 10^{2}$	$\mathbf{4.2330 \cdot 10^{2}}$
	30	$\mathbf{6.7292 \cdot 10^{2}}$	$6.7904 \cdot 10^{2}$	$7.1205 \cdot 10^{2}$	$7.3174 \cdot 10^{2}$	$6.9059 \cdot 10^{2}$	$\mathbf{6.8905 \cdot 10^{2}}$	$7.2329 \cdot 10^{2}$	$7.1081 \cdot 10^{2}$
	50	$1.4703 \cdot 10^{3}$	$\mathbf{1.4628 \cdot 10^{3}}$	$1.5537 \cdot 10^{3}$	$1.6018 \cdot 10^{3}$	$\mathbf{1.4518 \cdot 10^{3}}$	$1.4977 \cdot 10^{3}$	$1.6352 \cdot 10^{3}$	$1.5499 \cdot 10^{3}$
f_{10}	2	$5.2008 \cdot 10$	$\mathbf{4.9133 \cdot 10}$	$5.9181 \cdot 10$	$6.0217 \cdot 10$	$8.0638 \cdot 10$	$7.0828 \cdot 10$	$6.2908 \cdot 10$	$\mathbf{5.5046 \cdot 10}$
	10	$6.4561 \cdot 10^{2}$	$6.3973 \cdot 10^{2}$	$\mathbf{6.0526 \cdot 10^{2}}$	$6.1345 \cdot 10^{2}$	$\mathbf{6.2053 \cdot 10^{2}}$	$6.4226 \cdot 10^{2}$	$6.4227 \cdot 10^{2}$	$6.3270 \cdot 10^{2}$
	30	$1.9026 \cdot 10^{3}$	$1.7223 \cdot 10^{3}$	$1.5295 \cdot 10^{3}$	$\mathbf{1.2942 \cdot 10^{3}}$	$1.8724 \cdot 10^{3}$	$1.9307 \cdot 10^{3}$	$\mathbf{1.3993 \cdot 10^{3}}$	$1.4760 \cdot 10^{3}$
	50	$6.4178 \cdot 10^{3}$	$5.9230 \cdot 10^{3}$	$\mathbf{5.0535 \cdot 10^{3}}$	$5.1330 \cdot 10^{3}$	$6.1592 \cdot 10^{3}$	$6.1387 \cdot 10^{3}$	$\mathbf{4.7873 \cdot 10^{3}}$	$5.1577 \cdot 10^{3}$
		115	114	89	**82**	117	129	**69**	85

Source: produced by the author.

tests are conducted to distinguish the algorithms in terms of statistical significance.

To complement the above experiments, Table 16 summarizes the results with the following averages: CPU time (in seconds), number of iterations required to stop the algorithms, and success rate (in percentage). The values are all averaged based on $E = 134$ and also represents the average of all functions. As can be seen, the lower the dimension, the better the results. This point is clear when the test functions are set as $d = 2$. As the number of dimensions increases, even a well-behaved test function becomes complex to optimize. As a result, the success rate is approaching zero and the iteration number required to stop the algorithm is maximum. The versions Ψ^{late} are faster than Ψ^{fast} in all cases. This is due to the method to exchange information among particles. Ψ^{late} algorithms updates the global best position after a whole iteration is completed, i.e., only one verification of the swarm must be done per iteration. Thus, the particles move

Table 16 – Summary of performance criteria by grouping the results per dimension.

d	Criterion	$\xi = 10^{-2}$				$\xi = 10^{-8}$			
		Ψ_{var}^{late}	Ψ_{inv}^{late}	Ψ_{var}^{fast}	Ψ_{inv}^{fast}	Ψ_{var}^{late}	Ψ_{inv}^{late}	Ψ_{var}^{fast}	Ψ_{inv}^{fast}
2	sr	87.2388	**88.4328**	85.1492	85.2985	**70.8209**	70.22388	70	70.2985
	time	0.3274	**0.3144**	0.7605	0.7801	**0.5492**	0.5533	1.2488	1.2734
	iter.	1444.4832	**1405.51**	1445.1289	1477.7687	**2411.22**	2420.38	2429.64	2413.51
10	sr	15.7463	15.8209	**16.2687**	15.9701	8.806	15.4478	**15.8209**	15.5224
	time	**1.1523**	1.1534	2.612	2.6914	**1.2713**	1.2815	2.6576	2.7207
	iter.	4609.85	4605.85	**4593.57**	4598.7	4699.4	4685.72	**4675.23**	4675.84
30	sr	0	0	0	0	0	0	0	0
	time	1.8543	**1.8542**	3.643	3.717	**1.951**	1.9834	3.5915	3.6722
	iter.	5000	5000	5000	5000	5000	5000	5000	5000
50	sr	0	0	0	0	0	0	0	0
	time	2.8609	**2.8505**	4.6212	4.9515	3.1087	**2.9029**	4.5858	4.8319
	iter.	5000	5000	5000	5000	5000	5000	5000	5000

Source: produced by the author.

through the search space simultaneously. On the other hand, due to the fast exchange of information of Ψ^{fast}, a verification must be done after each particle's movement, i.e., the global best position can be updated multiple times at each iteration, which increases the computational time of the algorithm.

Although some algorithm are better than others regarding the success rate and number of iterations required to stop the algorithm, the winning algorithms are barely consistent. For example, the success rate of 88.43% with Ψ_{inv}^{late} is closely followed by 87.23% with Ψ_{var}^{late} in bi-dimensional problems with $\xi = 10^{-2}$. However, Ψ_{var}^{late} is better with 70.82% when $\xi = 10^{-8}$. In the 10-dimensional space, one can note that the best algorithm is Ψ_{var}^{fast}, independently of the error threshold predefined. The conclusions about the performance of the algorithms are compromised when 30- and 50-dimensional problems are evaluated, as the error thresholds were not reached. However, the authors decided to maintain these results since the computational times indicate the fastest algorithms.

In relation to the number of iterations required to stop the algorithm, the values follow the success rate, i.e., in 2-dimensional problems, Ψ_{inv}^{late} required less iterations to reach feasible solutions with $\xi = 10^{-2}$, while Ψ_{var}^{late} ended up with less iterations when $\xi = 10^{-8}$. When 10-dimensional problems are considered, the best algorithm is Ψ_{var}^{fast} in both error rates. The algorithms tie in high dimensional test functions.

As presented in Tables 15 and 16, it is quite difficult to distinguish the best algorithm, as some algorithm are better than others in some cases, but the same algorithm performed poorly in other scenarios. Since one of the goals of this paper is to indicate the best approach between Ψ_{var} and Ψ_{inv}, considering the methods to update the global memory, the next Section 4.2.3 provides a statistical hypothesis test based on the data from Table 15.

4.2.3 Friedman's and Wilcoxon's statistical tests with $k \times k$ and pair-wise comparisons

In this section, two well-known statistical hypotheses tests are conducted to find an indicative of the best PSO version based on the error rate described in Table 15: Friedman (FRIEDMAN, 1937) and Wilcoxon signed ranks (ZAR, 2007). The experiments assume the following null and alternative hypotheses in terms of algorithm results[1]:

- H_0: both algorithms have no significant difference;
- H_1: both algorithms have significant difference.

The first part of the statistical hypothesis analysis evaluates Friedman's test and post-hoc procedure to decide whether exist statistical significance among the performances of the algorithm. Friedman's test is used for multiple comparisons. In the context of this thesis, multiple comparisons is related to each pair of possible comparison between two algorithms ($k \times k$) by considering four PSO versions ($k = 4$) and a family of different hypotheses $h = \frac{k(k-1)}{2}$. In the case of statistical significance among algorithms, a post-hoc procedure is performed to find the p-values which determines the degree of rejection of each hypothesis. Furthermore, the p-values are adjusted to consider the family error accumulated (Family-Wise Error Rate – FWER), which is the probability of making one or more false discoveries among all the hypotheses when performing multiple pairwise tests.

The second part of the statistical hypothesis analysis investigates Wilcoxon's test to provide specific insights between two pairs of comparisons: 1) Ψ_{var} *versus* Ψ_{inv} ; 2) Ψ^{late} *versus* Ψ^{fast} . The goal of the second statistical hypothesis test is to validate and confirm the findings of Friedman's test.

Friedman's test is a non-parametric statistical hypothesis test used to detect differences in the median values at least between two populations. In the context of algorithms' performance, Freedman test detects whether there are significance differences between two or more algorithms (DERRAC et al., 2011). If the statistical significance holds for at least two algorithms under consideration, a post-hoc procedure needs to be executed to find the specific algorithms where the significance exists. To compute the Friedman statistic, the original results from Table 15 must be converted through a ranking-based transformation (HOLLANDER; WOLFE; CHICKEN, 2013). For each problem i, rank the results of each algorithm j from 1 (best result) to k (worst result) as $r_{ij}, 1 \leq i \leq m$, and $1 \leq j \leq k$. After that, the ranks of each algorithm has to be averaged: $R_j = \frac{1}{m} \sum_{i=1}^{m} r_{ij}$.

[1] Generic null and alternative hypotheses concern the type of statistical measure of population for Friedman's and Wilcoxon's tests, respectively, median and mean.

Finally, each average is used to compute the Friedman statistic

$$F_f = \frac{12m}{k(k+1)}\left(\sum_{j=1}^{k} R_j^2 - \frac{k(k+1)^2}{4}\right), \tag{4.5}$$

where m stands for the number of problems and k is the number of algorithms. When $m > 10$ and $k > 4$, F_f follows a χ^2 distribution with $k-1$ degrees of freedom. For a smaller number of algorithms and problems, the critical values can be extracted from the Friedman χ_F^2 distribution table (ZAR, 2007).

Before proceeding with the Friedman's test, Table 17 describes the average ranks of each PSO version obtained through the application of ranking-based transformation. When $\xi = 10^{-2}$, the results indicates the version Ψ_{inv}^{fast} as the best performing algorithm of the comparison, although the other approaches are close to each other where the maximum difference of average ranks is 0.825 (2.875 - 2.05). Thus, the sum of ranks are more likely to be equal among the algorithms. Whilst the algorithm Ψ_{var}^{fast} is the best approach when $\xi = 10^{-8}$. However, the maximum difference of average ranks is 1.5 (3.225 - 1.725). Therefore, there is a strong inclination for good performance for the algorithm Ψ_{var}^{fast}. In general, until this simple analysis, the conclusion is that the fast information exchange has an important role in the search process, but the rotational variance (or invariance) property must be analysed more carefully, once a tie was found.

Table 17 – Average ranks achieved by the Friedman's test using Table 15.

Algorithm	$\xi = 10^{-2}$	Order	$\xi = 10^{-8}$	Order
Ψ_{var}^{late}	2.875	4	2.925	3
Ψ_{inv}^{late}	2.850	3	3.225	4
Ψ_{var}^{fast}	2.225	2	1.725	1
Ψ_{inv}^{fast}	2.050	1	2.125	2

Source: produced by the author.

Table 18 highlights the Friedman statistics, p-values and critical value. The test statistic is greater than the critical value for both error thresholds. Also, the p-value is smaller than the chosen significance level ($\alpha = 0.05$), which suggests there is at least one significant difference among the algorithms under consideration.

After the whole multiple comparison analysis and the strong evidence of significant difference among the algorithms' performance, the post-hoc procedure must be taking into account. A family of hypothesis, in this case $h = 6$ hypothesis, is defined to test each possible pairs of comparisons between algorithms. The p-value of every hypothesis between the algorithm i and j is obtained through the conversion of the average ranks of each algorithm by using a normal approximation. The z-values are used to find the corresponding probability (p-value) from the standard normal distribution $\mathcal{N}(0,1)$. The

Table 18 – Friedman statistics and related p-values are shown considering both error thresholds. The critical value is obtained through a Chi-squared with *confidence level* $= 1 - \alpha$ ($\alpha = 0.05$) and $v = k - 1$ ($k = 4$) degrees of freedom.

	$\xi = 10^{-2}$	$\xi = 10^{-8}$
Friedman statistic	12.99	34.74
p-value	$4.6583 \cdot 10^{-3}$	$1.3825 \cdot 10^{-7}$
Critical value ($\chi^2_{0.95,3}$)	7.8147	

Source: produced by the author.

Friedman z-value is computed as

$$z_{ij} = \frac{R_i - R_j}{\sqrt{\frac{k(k+1)}{6n}}}. \tag{4.6}$$

The unadjusted p-values are shown in the Table 19. These p-values are not adequate to be directly compared with the significance level due to the FWER[2]. To compensate the family error accumulated, adjusted p-values are used to directly compare the values with the significance level. In this thesis, three adjustment procedures are provided: Nemenyi, Holm, and Shaffer. More information about each of the procedures can be found in (DERRAC et al., 2011). From Table 19, the $(*)$ symbol indicates which algorithm is better than other. As previously observed, all PSO versions equipped with the fast information exchange have better results with statistical significance. The smaller the p-value, the stronger the evidence against H_0. As can be seen, the statistical significance in this scenario seems to be more related to the way the information flow among the particles rather than the manner the random variables are handling in the velocity update equation.

Table 19 – Friedman p-values and adjusted p-values for multiple comparisons among all algorithms.

Hyphotesis	$\xi = 10^{-2}$				$\xi = 10^{-8}$			
	Unadjusted p	Nemenyi	Holm	Shaffer	Unadjusted p	Nemenyi	Holm	Shaffer
Ψ^{late}_{var} *vs* Ψ^{late}_{inv}	0.9310	1	1	1	0.2987	1	0.3317	0.3317
Ψ^{late}_{var} *vs* Ψ^{fast*}_{var}	0.0243	0.1461	0.0974	0.0730	$3.2256 \cdot 10^{-5}$	$\mathbf{1.9354 \cdot 10^{-4}}$	$\mathbf{1.6128 \cdot 10^{-4}}$	$\mathbf{9.6769 \cdot 10^{-5}}$
Ψ^{late}_{var} *vs* Ψ^{fast*}_{inv}	0.0043	**0.0256**	**0.0256**	**0.0256**	0.0056	**0.0335**	**0.0168**	**0.0168**
Ψ^{late}_{inv} *vs* Ψ^{fast*}_{var}	0.0304	0.1823	0.0974	0.0911	$2.0346 \cdot 10^{-7}$	$\mathbf{1.2207 \cdot 10^{-6}}$	$\mathbf{1.2207 \cdot 10^{-6}}$	$\mathbf{1.2207 \cdot 10^{-6}}$
Ψ^{late}_{inv} *vs* Ψ^{fast*}_{inv}	0.0056	**0.0335**	**0.0279**	**0.0256**	$1.3868 \cdot 10^{-4}$	$\mathbf{8.3208 \cdot 10^{-4}}$	$\mathbf{5.5472 \cdot 10^{-4}}$	$\mathbf{4.1604 \cdot 10^{-4}}$
Ψ^{fast}_{var} *vs* Ψ^{fast}_{inv}	0.5444	1	1	1	0.1659	0.9951	0.3317	0.3317

Source: produced by the author.

The second part of the statistical hypothesis analysis evaluates two pairs of comparisons: 1) Ψ_{var} *vs* Ψ_{inv} ; 2) Ψ^{late} *vs* Ψ^{fast} . The first comparison investigates whether the

[2] The Family-Wise Error Rate for multiple comparisons can be computed as $1 - (1 - \alpha)^h$. Thus, the probability of making one or more Type I error is 26.5, which is quite high.

rotational information influences the performance of the algorithms when both exchanges of information are associated with the same algorithm. The second comparison verifies whether the type of information exchange benefits somehow the results obtained by considering the union of the rotational information. The following unions regarding the results presented in Table 15 are considered:

$$\begin{aligned}
\Psi_{var} &= \Psi_{var}^{late} \cup \Psi_{var}^{fast} \\
\Psi_{inv} &= \Psi_{inv}^{late} \cup \Psi_{inv}^{fast} \\
\Psi^{late} &= \Psi_{var}^{late} \cup \Psi_{inv}^{late} \\
\Psi^{fast} &= \Psi_{var}^{fast} \cup \Psi_{inv}^{fast}
\end{aligned}$$

A non-parametric test, such as Wilcoxon's test, aims to detect significant differences between two sample means and can be applied to continuous data through a ranking-based transformations (HOLLANDER; WOLFE; CHICKEN, 2013). Herein, rank is to assign values to samples from 1 to m (number of problems). Equations 4.7 and 4.8 define the sum of ranks based on the difference d_i between the performance scores of two algorithms on ith out of m problems, both described as

$$R^+ = \sum_{d_i>0} \text{rank}(d_i) + \frac{1}{2} \sum_{d_i=0} \text{rank}(d_i) \tag{4.7}$$

and

$$R^- = \sum_{d_i<0} \text{rank}(d_i) + \frac{1}{2} \sum_{d_i=0} \text{rank}(d_i). \tag{4.8}$$

The differences are ranked according to their absolute values. The rank of tie cases (when two or more equal d_i values are found) are replaced by the average rank that would originally be assigned to $d_{k,\ldots,l}$ (such that $d_k = d_{k+1} = \ldots = d_l$, with $k < l$). Ranks of $d_i = 0$ are split evenly among the sums, ignoring one difference if the total is an odd number. The variable R^+ is the sum of positive ranks for the problems whose algorithm 1 surpassed the algorithm 2, and R^- is the sum of ranks for the opposite. The goal is to count the number of cases in which an algorithm is the overall winner.

The table of reference for critical values for the two-tailed of the Wilcoxon T distribution is available in (ZAR, 2007). Note that, the authors consider to consult critical values up to the maximum number of problems $m \leq 50$. For $m > 50$, a procedure to compute the p-value is performed. Table 20 exhibits the statistical hypothesis test for variant and invariant PSO versions. The number of optimization problem is $m = 80$. Thus, the p-value is considered in this case to reject or not H_0, instead of searching for critical values in the table of reference. There is no statistical difference between the rotationally variant and invariant PSO versions, although the particular results of Ψ_{var} are a bit better.

This indicates that from the statistical point of view, H_0 must be accepted as true, as the minimum values for the levels of significance to reject H_0 should be 84.79% ($\xi = 10^{-2}$) and 17.32% ($\xi = 10^{-8}$). Therefore, it is expected that the variant and invariant versions of PSO provide similar results when applied in optimization problems.

Table 20 – Wilcoxon signed ranks test results with significance level $\alpha = 0.05$. The results of four PSO versions were grouped into two: variant and invariant.

Hypothesis	$\xi = 10^{-2}$			$\xi = 10^{-8}$		
	R^+	R^-	p	R^+	R^-	p
Ψ_{var} *vs* Ψ_{inv}	**1660**	1580	0.8479	**1904**	1336	0.1732

Source: produced by the author.

Finally, the second result related to the comparison of two algorithms is available in Table 21, where Ψ^{late} and Ψ^{fast} are under consideration, grouping all dimensional problems along with late and fast PSO versions, again with $m = 80$. The overall analysis shows Ψ^{fast} as the best version of PSO, regardless of the rotation (or not) of a particle motion. As expected, the null hypothesis is rejected with a very low probability of making a Type I error, i.e., a false-positive indication of statistical significance between the algorithms. This significance difference confirms what was observed by the Friedman's test in the Table 18 where the fast information flow, i.e., updating the global best position soon after the movement of a particle, is rather preferable than after a complete iteration.

Table 21 – Wilcoxon signed ranks test results with significance level $\alpha = 0.05$. The results of four PSO versions were grouped into two: late and fast.

Hypothesis	$\xi = 10^{-2}$			$\xi = 10^{-8}$		
	R^+	R^-	p	R^+	R^-	p
Ψ^{late} *vs* Ψ^{fast*}	692	**2548**	$\mathbf{8.5489 \cdot 10^{-6}}$	496	**2744**	$\mathbf{7.0056 \cdot 10^{-8}}$

Source: produced by the author.

4.3 Results: ISAPSO algorithm

Since the methodology to evaluate a new algorithm is already established by both previously Sections 4.1 and 4.2, this section evaluates ISAPSO algorithm by applying it in a set of optimization problems. The idea of evaluating the algorithm in test functions are based on SAPSO algorithm in which ten benchmark optimization problems were under consideration. However, in this context, ISAPSO algorithm is performed in CEC 2017 benchmark suite presented in Table 13. Those optimization problems are difficult to optimize, as all functions are rotated, translated, and some of them are composition

functions. All numerical simulations are based on the minimum number of experiments found in Section 3.3.4, which is $E = 55$.

4.3.1 Numerical simulation on benchmark functions

To accomplish this experiments, ISAPSO algorithm is compared with seven other PSO variants strictly related to the approach, also including SAPSO algorithm. The test functions are predefined with 2, 10, and 30 number of dimensions. The stop criterion in all function is $\epsilon = 10^{-8}$. The maximum number of iteration is $T = 1000$ for all algorithms and their parameter settings are the ones presented in Appendix B in the Table 28. ISAPSO algorithm defined the inertia weight $w = 0.7298$, the social coefficient $c_1 = 1.4962$ and the gradient coefficient $c_2 = 10^{-2}$ in all test functions. The angle of rotation is based on WPSO algorithm (WILKE; KOK; GROENWOLD, 2007b), which is $\alpha = 3$. The lower (d_{low}) and upper (d_{high}) bounds for the diversity control are, respectively, 10^{-2} and 0.25 for all test functions.

Since most optimization problems have different global minimum value, Table 22 shows the average error rate obtained by the difference between the known global minimum value and the value found by each algorithm. In this form, the results are in accordance with *the lower the value the better the results*. The last line is simply the sum of ranks of each algorithm that goes from 1 to 8.

Table 22 – Average error rate obtained from CEC 2017 benchmark functions.

f_i	d	ARPSO	GPSO	DGHPSOGS	SPSO	WPSO	BPSO	SAPSO	ISAPSO
f_1	2	$1.4831 \cdot 10^{3}$	$\mathbf{8.4579 \cdot 10^{-8}}$	$1.4034 \cdot 10^{3}$	$9.7074 \cdot 10^{2}$	$5.9158 \cdot 10^{-8}$	$2.4165 \cdot 10^{-2}$	$1.6252 \cdot 10^{1}$	$6.0512 \cdot 10^{-1}$
	10	$1.5985 \cdot 10^{9}$	$\mathbf{8.9200 \cdot 10^{-9}}$	$1.7853 \cdot 10^{8}$	$5.1574 \cdot 10^{3}$	$7.0624 \cdot 10^{3}$	$2.0202 \cdot 10^{4}$	$2.4202 \cdot 10^{3}$	$1.2215 \cdot 10^{3}$
	30	$2.5758 \cdot 10^{10}$	$\mathbf{9.9116 \cdot 10^{2}}$	$9.4853 \cdot 10^{9}$	$6.4100 \cdot 10^{3}$	$5.5135 \cdot 10^{3}$	$1.1662 \cdot 10^{6}$	$5.0265 \cdot 10^{3}$	$9.5499 \cdot 10^{4}$
f_2	2	$4.6094 \cdot 10^{-9}$	$8.3569 \cdot 10^{-6}$	$4.4206 \cdot 10^{-9}$	$4.8907 \cdot 10^{-9}$	$5.0468 \cdot 10^{-9}$	$4.0575 \cdot 10^{-7}$	$2.1367 \cdot 10^{-8}$	$\mathbf{1.9436 \cdot 10^{-9}}$
	10	$6.8008 \cdot 10^{3}$	$2.1860 \cdot 10^{-8}$	$7.3862 \cdot 10^{3}$	$9.1150 \cdot 10^{-9}$	$8.8266 \cdot 10^{-9}$	$4.4078 \cdot 10^{-2}$	$6.9645 \cdot \mathbf{10^{-9}}$	$\mathbf{4.5218 \cdot 10^{-9}}$
	30	$1.4756 \cdot 10^{5}$	$\mathbf{1.5524 \cdot 10^{-5}}$	$1.2845 \cdot 10^{5}$	$9.0519 \cdot 10^{4}$	$3.5981 \cdot 10^{4}$	$6.6638 \cdot 10^{3}$	$3.0443 \cdot 10^{4}$	$4.3111 \cdot 10^{3}$
f_3	2	$4.1464 \cdot 10^{-9}$	$2.8405 \cdot 10^{-9}$	$4.9634 \cdot 10^{-9}$	$6.1877 \cdot 10^{-9}$	$4.4528 \cdot 10^{-9}$	$4.8064 \cdot 10^{-9}$	$5.6839 \cdot 10^{-9}$	$\mathbf{1.9121 \cdot 10^{-9}}$
	10	$1.1774 \cdot 10^{2}$	$7.2483 \cdot 10^{-2}$	$3.8570 \cdot 10^{1}$	5.9572	$7.2485 \cdot 10^{-2}$	2.0223	$3.4765 \cdot 10^{-2}$	$\mathbf{3.8670 \cdot 10^{-3}}$
	30	$3.4832 \cdot 10^{3}$	**1.6671**	$1.0902 \cdot 10^{3}$	$7.2141 \cdot 10^{1}$	$5.5267 \cdot 10^{1}$	$6.1164 \cdot 10^{1}$	$1.6062 \cdot 10^{1}$	$1.0899 \cdot 10^{2}$
f_4	2	$5.5079 \cdot 10^{-2}$	$3.3213 \cdot 10^{-4}$	$4.5279 \cdot 10^{-9}$	$9.0451 \cdot 10^{-2}$	$4.1607 \cdot 10^{-1}$	$5.2461 \cdot 10^{-1}$	$1.1164 \cdot 10^{-1}$	$\mathbf{4.3368 \cdot 10^{-9}}$
	10	$4.2577 \cdot 10^{1}$	$3.7622 \cdot 10^{1}$	$2.8569 \cdot 10^{1}$	$3.4118 \cdot 10^{1}$	$4.8029 \cdot 10^{1}$	$5.4968 \cdot 10^{1}$	$4.7409 \cdot 10^{1}$	$\mathbf{1.5947 \cdot 10^{1}}$
	30	$3.1996 \cdot 10^{2}$	$2.8611 \cdot 10^{2}$	$1.8180 \cdot 10^{2}$	$1.8743 \cdot 10^{2}$	$2.2808 \cdot 10^{2}$	$2.5188 \cdot 10^{2}$	$2.1658 \cdot 10^{2}$	$\mathbf{1.6544 \cdot 10^{2}}$
f_5	2	$4.6972 \cdot 10^{-9}$	$7.7109 \cdot 10^{-9}$	$5.1564 \cdot 10^{-9}$	$5.2709 \cdot 10^{-9}$	$4.6474 \cdot 10^{-9}$	$7.5340 \cdot 10^{-8}$	$\mathbf{4.2638 \cdot 10^{-9}}$	$4.9434 \cdot 10^{-9}$
	10	$3.3666 \cdot 10^{2}$	$2.4937 \cdot 10^{2}$	$1.4763 \cdot 10^{2}$	$1.6731 \cdot 10^{2}$	$2.1299 \cdot 10^{2}$	$3.8567 \cdot 10^{2}$	$1.3070 \cdot 10^{2}$	$\mathbf{1.6614 \cdot 10^{-1}}$
	30	$7.2878 \cdot 10^{3}$	$8.8381 \cdot 10^{3}$	$5.3240 \cdot 10^{3}$	$4.0702 \cdot 10^{3}$	$3.8335 \cdot 10^{3}$	$5.0295 \cdot 10^{3}$	$\mathbf{3.4222 \cdot 10^{3}}$	$4.5183 \cdot 10^{3}$
f_6	2	$1.8234 \cdot 10^{1}$	2.9805	3.1759	$2.0555 \cdot 10^{1}$	$1.6185 \cdot 10^{1}$	$2.0890 \cdot 10^{1}$	$1.2878 \cdot 10^{1}$	**2.7093**
	10	$1.1200 \cdot 10^{3}$	$1.0958 \cdot 10^{3}$	$9.4428 \cdot 10^{2}$	$9.5647 \cdot 10^{2}$	$1.1440 \cdot 10^{3}$	$1.1519 \cdot 10^{3}$	$1.2696 \cdot 10^{3}$	$\mathbf{8.9359 \cdot 10^{2}}$
	30	$6.0677 \cdot 10^{3}$	$5.0655 \cdot 10^{3}$	$4.9758 \cdot 10^{3}$	$4.2108 \cdot 10^{3}$	$4.3209 \cdot 10^{3}$	$4.2840 \cdot 10^{3}$	$4.3205 \cdot 10^{3}$	$\mathbf{3.8937 \cdot 10^{3}}$
f_8	2	$5.0338 \cdot 10^{-5}$	4.9521	$2.6008 \cdot 10^{-5}$	$\mathbf{7.0898 \cdot 10^{-9}}$	$1.0909 \cdot 10^{1}$	$5.5556 \cdot 10^{-3}$	3.6365	$1.9640 \cdot 10^{-5}$
	10	$9.4641 \cdot 10^{2}$	$6.0381 \cdot 10^{2}$	$7.3807 \cdot 10^{2}$	$5.4368 \cdot 10^{2}$	$6.0921 \cdot 10^{2}$	$7.2021 \cdot 10^{2}$	$5.2273 \cdot 10^{2}$	$\mathbf{5.1094 \cdot 10^{2}}$
	30	$5.7795 \cdot 10^{3}$	$3.8238 \cdot 10^{3}$	$4.5192 \cdot 10^{3}$	$3.5250 \cdot 10^{3}$	$4.0518 \cdot 10^{3}$	$5.1052 \cdot 10^{3}$	$\mathbf{2.9448 \cdot 10^{3}}$	$3.5152 \cdot 10^{3}$
f_9	2	$\mathbf{6.9466 \cdot 10^{-3}}$	5.4096	3.4442	$1.8190 \cdot 10^{1}$	7.6491	7.3423	5.8938	$2.4780 \cdot 10^{-2}$
	10	$4.4195 \cdot 10^{2}$	$3.9990 \cdot 10^{2}$	$4.2716 \cdot 10^{2}$	$4.0111 \cdot 10^{2}$	$4.7874 \cdot 10^{2}$	$4.5912 \cdot 10^{2}$	$4.6193 \cdot 10^{2}$	$\mathbf{3.9647 \cdot 10^{2}}$
	30	$8.3421 \cdot 10^{2}$	$5.0770 \cdot 10^{2}$	$6.8814 \cdot 10^{2}$	$5.2179 \cdot 10^{2}$	$\mathbf{5.0062 \cdot 10^{2}}$	$5.1275 \cdot 10^{2}$	$8.3483 \cdot 10^{2}$	$5.8442 \cdot 10^{2}$
f_{10}	2	$6.8253 \cdot 10^{1}$	$7.1834 \cdot 10^{1}$	$7.7605 \cdot 10^{1}$	$1.1818 \cdot 10^{2}$	$1.0643 \cdot 10^{2}$	$1.3008 \cdot 10^{2}$	$7.2803 \cdot 10^{1}$	$\mathbf{4.0000 \cdot 10^{1}}$
	10	$6.5113 \cdot 10^{2}$	$4.4980 \cdot 10^{2}$	$6.1151 \cdot 10^{2}$	$\mathbf{4.2060 \cdot 10^{2}}$	$4.8300 \cdot 10^{2}$	$4.6524 \cdot 10^{2}$	$6.4834 \cdot 10^{2}$	$4.4884 \cdot 10^{2}$
	30	$2.2296 \cdot 10^{3}$	$3.6839 \cdot 10^{2}$	$1.5455 \cdot 10^{3}$	$4.6973 \cdot 10^{2}$	$5.0001 \cdot 10^{2}$	$4.6358 \cdot 10^{2}$	$4.2966 \cdot 10^{2}$	$\mathbf{1.0459 \cdot 10^{2}}$
		165	103	136	118	128	155	112	**55**

Source: produced by the author.

In general, ISAPSO algorithm has obtained the lower error rates in most optimization problems. Although it can be considered a naive evaluation, this performance rendered the lower sum of rank for the proposed approach, providing better results even when compared with GPSO and SAPSO algorithms, respectively second and third place. Further evaluations, regarding the statistical hypothesis testing, are provided to certify whether there is statistical significance in the comparisons.

Table 23 summarizes other performance criteria, such as success rate, computational time and minimum number required to stop the algorithm, to analyze. Clearly, ISAPSO algorithm has the best average success rate when optimization problems are set as 2-dimensions. However, for 10-dimensions, GPSO algorithm has higher success rate than ISAPSO algorithm. Another important consideration is that the ISAPSO algorithm is faster than SAPSO algorithm. In ISAPSO algorithm, the gradient calculus is performed just when required, i.e., when $I_i = 0$, which in turn speed up the execution of the algorithm. The fastest algorithm among the PSOs under evaluation is the ARPSO algorithm. This algorithm has no gradient calculus and no rotation matrix to build, which contributes to be executed faster. In contrast, ARPSO algorithm holds the worst accuracy results in average (the sum of rank is 165 from Table 22). GPSO and WPSO algorithms usually stop sooner than the other algorithms. This can be seen, respectively, when the functions have 2-dimensions and 10-dimensions. No further evaluations can be made for higher dimensions, since no algorithm has stopped before the maximum number of iterations has been reached.

Table 23 – Summary of performance criteria: success rate, computational time and minimum number of iterations to stop the algorithm. The results are grouped per dimensions.

d	**Criterion**	**ARPSO**	**GPSO**	**DGHPSOGS**	**SPSO**	**WPSO**	**BPSOb**	**SAPSO**	**ISAPSO**
2	sr	53.3334	57.3723	53.7378	58.7878	57.9811	12.1211	46.6667	**61.9189**
	time	**0.07668**	1.09898	0.305	0.2428	0.2688	1.0416	0.7778	0.5873
	iter.	551.6747	**451.6388**	667.9838	481.4263	462.8187	919.8525	672.099	613.3313
10	sr	0	**24.8478**	0	11.3133	14.95	0	1.4144	11.3133
	time	**0.1517**	2.3475	1.3758	0.5011	0.6209	1.2062	2.1038	1.4052
	iter.	1000	**790.9414**	1000	939.0687	932.7798	1000	997.8566	949.9515
30	sr	0	0	0	0	0	0	0	0
	time	**0.2676**	6.4561	6.8874	0.652	1.0846	1.5711	7.6602	5.7626
	iter.	1000	1000	1000	1000	1000	1000	1000	1000

Source: produced by the author.

4.3.2 Friedman's statistical test with $1 \times k$ comparisons

Statistical inference described in this section is heavily based on Section 4.2.3. The main goal is to investigate whether the results presented in Table 22 has statistical significance in the context of error rates. Again, Friedman's test is used as a statistical

hypothesis test in the following experiments. First of all, Table 24 shows the average ranks of each PSO version obtained by the ranking-based transformation. ISAPSO algorithm comes in first place, while GPSO and SAPSO come in second and third place, respectively. These ranks are used to perform the Friedman statistic illustrated in Table 25.

Table 24 – Average ranks achieved by the Friedman's test using Table 22.

Algorithm	$\xi = 10^{-8}$	Order
ISAPSO	2.037	1
GPSO	3.8148	2
SAPSO	4.1481	3
SPSO	4.3704	4
WPSO	4.7407	5
DGHPSOGS	5.037	6
BPSO	5.7407	7
ARPSO	6.1111	8

Source: produced by the author.

Table 25 shows the Friedman statistic, p-value and critical value. As one can infer, Friedman statistic is above critical value. Also, p-value is smaller than the chosen significance level ($\alpha = 0.05$). This indicates that there is at least one significant difference among the algorithms' results.

Table 25 – Friedman statistics and related p-values are shown considering both error thresholds. The critical value is obtained through a Chi-squared with *confidence level* $= 1 - \alpha$ ($\alpha = 0.05$) and $v = k - 1$ ($k = 4$) degrees of freedom.

	$\xi = 10^{-8}$
Friedman statistic	50.2099
p-value	$1.3137 \cdot 10^{-8}$
Critical value ($\chi^2_{0.95,7}$)	14.0671

Source: produced by the author.

A post-hoc procedure to find the statistical significance between the algorithms is taking into account. The type of post-hoc procedure used here is $1 \times k$ (in contrast with $k \times k$ method used earlier), i.e., there is a control method (ISAPSO algorithm) that is compared with $k - 1$ algorithms (excluding the control method). In this case, a family of $h = 7$ hypothesis is defined to test each pair of possible comparisons. The goal of this type of post-hoc procedures is to test whether a newly proposed algorithm is better than existing ones.

The unadjusted p-values are presented in Table 26. As previously stated, these p-values are not suitable to be compared directly with the significance level, since they

Table 26 – Friedman p-values and adjusted p-values by considering ISAPSO algorithm as the control method.

Algorithm	Unadjusted p	Bonferroni-Dunn	Holm	Hockberb	Hommel	Holland	Finner	Li
SAPSO	$1.5420 \cdot 10^{-3}$	$1.0794 \cdot 10^{-2}$	$3.0839 \cdot 10^{-3}$	$3.0839 \cdot 10^{-3}$	$3.0839 \cdot 10^{-3}$	$3.0816 \cdot 10^{-3}$	$1.7987 \cdot 10^{-3}$	$1.5396 \cdot 10^{-3}$
ARPSO	$9.8940 \cdot 10^{-10}$	$6.9258 \cdot 10^{-9}$	$6.9258 \cdot 10^{-9}$	$6.9258 \cdot 10^{-9}$	$6.9258 \cdot 10^{-9}$	$6.9258 \cdot 10^{-9}$	$6.9258 \cdot 10^{-9}$	$9.8940 \cdot 10^{-10}$
GPSO	$7.6608 \cdot 10^{-3}$	$5.3625 \cdot 10^{-2}$	$7.6608 \cdot 10^{-3}$	$7.6608 \cdot 10^{-3}$	$7.6608 \cdot 10^{-3}$	$7.6608 \cdot 10^{-3}$	$7.6608 \cdot 10^{-3}$	$7.6025 \cdot 10^{-3}$
DGHPSOGS	$6.7953 \cdot 10^{-6}$	$4.7567 \cdot 10^{-5}$	$3.3977 \cdot 10^{-5}$	$3.3977 \cdot 10^{-5}$	$3.3977 \cdot 10^{-5}$	$3.3976 \cdot 10^{-5}$	$1.5856 \cdot 10^{-5}$	$6.7953 \cdot 10^{-6}$
SPSO	$4.6526 \cdot 10^{-4}$	$3.2568 \cdot 10^{-3}$	$1.3958 \cdot 10^{-3}$	$1.3958 \cdot 10^{-3}$	$1.3958 \cdot 10^{-3}$	$1.3951 \cdot 10^{-3}$	$6.5130 \cdot 10^{-4}$	$4.6504 \cdot 10^{-4}$
WPSO	$5.0015 \cdot 10^{-5}$	$3.5011 \cdot 10^{-4}$	$2.0006 \cdot 10^{-4}$	$2.0006 \cdot 10^{-4}$	$2.0006 \cdot 10^{-4}$	$2.0005 \cdot 10^{-4}$	$8.7525 \cdot 10^{-5}$	$5.0013 \cdot 10^{-5}$
BPSO	$2.7673 \cdot 10^{-8}$	$1.9371 \cdot 10^{-7}$	$1.6604 \cdot 10^{-7}$	$1.6604 \cdot 10^{-7}$	$1.6604 \cdot 10^{-7}$	$1.6604 \cdot 10^{-7}$	$9.6856 \cdot 10^{-8}$	$2.7673 \cdot 10^{-8}$

Source: produced by the author.

inherits the family-wise error rate due to multiple comparisons (family of hypothesis). The probability of making one or more Type I error is 30.17%, which is rather high. Thus, the adjusted p-values can deal with this problem by considering the family error accumulated and can be compared directly with the significance level chosen. Table 26 also provides seven algorithms for adjustment of p-values. One can see, even after the adjustment of p-values, ISAPSO algorithm shows a significant improvement over all other PSO algorithms. Li's procedure provides the most powerful behavior, reaching the lowest p-values in the comparisons.

5 Conclusion

This thesis presented enhancements for the PSO algorithm by improving its global search mechanism. The first improvement was the SAPSO algorithm, a semi-autonomous particle swarm optimizer, which uses the gradient information along with attractive and repulsive scheme to reduce computational efforts of local investigation and avoids local optima as a final result. In the proposed algorithm, each particle has the autonomy to decide between follow its own negative gradient direction or follow the global tendency of the swarm, i.e., the global best position found so far. This decision is taken individually by each particle at each iteration of the search process. The second improvement yielded a new version of SAPSO algorithm, which was called ISAPSO. Unlike the dynamic parameters of SAPSO algorithm for each test function, ISAPSO algorithm used static parameters, regarding social and gradient coefficients, and a new velocity update equation that embodied a rotation matrix to face the lacking of directional diversity. ISAPSO algorithm was a result of the empirical analysis on rotation and information exchange among particles. In this numerical simulation, the goal was to discover whether rotation and information exchange affects the performance of classical PSO versions. Additionally, a new method to select a minimum number of executions based on the Law of Large Number was also provided.

Regarding the SAPSO algorithm, the particles are called *semi-autonomous* which has a relation to the internal decision taken by each particle of the swarm, and the attractive and repulsive schemes. SAPSO algorithm starts with each particle investigating its own area near to its initial position in the search space, i.e., the swarm is in the attraction phase at first. When the swarm loses its diversity, it switches to the repulsion phase and each particle starts to follow the opposite direction of the global tendency. This behavior permits the swarm to avoid a local minimum as a final result and also prevents premature convergence. When the diversity is recovered, i.e., the diversity value is above an upper threshold, the whole swarm turns to the attraction phase again, thus each particle uses the negative gradient direction, searching for other promising areas. This partial autonomy given to the particles has brought promising results when applied in multimodal functions, specially when compared to PSOs with similar behavior of attraction and repulsion, such as ARPSO and DGHPSOGS.

The numerical results were provided to certify that the proposed method can reach at least the same performance of other well-known PSOs. The results showed that the proposed SAPSO algorithm provided better results regarding the findings of a global minimum and fine-tuning of the final solution, although higher-dimensional problems (above 30 dimensions) has been time-consuming due to the use of gradient-based information by

each particle, which in turns can be considered as one of the the current limitations of SAPSO (consequently, ISAPSO algorithm as well). Despite this, the proposed approach is considered the most reliable algorithm to find the global minimum of multimodal test functions. The analysis was conducted on the application of the SAPSO algorithm and other three PSOs on the De Jong's benchmark optimization problems. SAPSO algorithm showed to perform better for a variety of test functions and different dimensional settings. Due to a similar performance of SAPSO and GPSO algorithms when the dimensions were set as 10, 20 and 30, other tests were applied in even higher dimensional settings. When the SAPSO and GPSO algorithms were applied in multimodal functions with 60 to 150 dimensions, the SAPSO algorithm showed to perform significantly better in four out of five multimodal test functions. This result is expected due to the "no free lunch theorem". Besides, the SAPSO algorithm provides features to avoid local optima and premature converge.

Experimental tests were performed and analysed to ensure that the diversity control mechanism plays an important role in the avoidance of local minima. The proposed algorithm kept the diversity of the swarm adaptively in the search process when compared to ARPSO and DGHPSOGS algorithms. In addition, the analysis has shown that the SAPSO algorithm applied a proper exploration and exploitation trade-off, using the repulsion phase with more accuracy than the other PSOs. The proposed algorithm used less repulsion phases during the search process of the global minimum, due to the smooth decrease of the diversity value right after a repulsion phase occurs. Even with less repulsion phases, the SAPSO algorithm was able to properly explore the search space and still find the global minimum in all test functions.

This thesis also presented a discussion about rotation and information exchange mechanism of PSO. Throughout the work, a methodology to make multiple comparisons among classical versions of PSO algorithm based on Clerc's rules was performed. Rule 1 was covered by the shifted and rotated features applied in each test function. Rule 2 was fulfilled by the method of selecting the minimum number of executions to provide stable and reliable results for further analyses. The experiments assume multiple performance criteria to evaluate the PSO versions, thus Rule 3 was accomplished. Finally, Rules 4 and 5 were carried out by floating-point numbers with double-precision of 64 bits and a Mersenne Twister library, respectively.

A method to estimate a minimum number of executions based on the characteristics of each optimization problem and the performance of the algorithm was proposed. The method assumed the success rate as the performance measure of the algorithm in relation to the optimization problems. When the success rate value is close enough to the average success rate value, the proposed algorithm stops and reports the number of executions required to stabilize the curve. If more than one optimization problem is under evaluation,

the average number of executions can be considered. All experiments of this thesis were based on the number of executions required to find stable results in multiple optimization problems.

Experiments were carried out on multiple optimization problems with different dimensional scenarios to evaluate four PSO versions. Two non-parametric hypothesis tests revealed no statistical significance between the rotationally variant and invariant versions of PSO. However, the rotationally variant version was the overall winner in the sum of ranks, regarding the Wilcoxon signed test. On the other hand, when the PSO versions use a fast information exchange among particles, the null hypothesis was rejected with a very low probability. However, the late information exchange among particles was computationally faster than the fast information exchange version. Although rotation variance is related to the bias phenomenon in PSO algorithm, to achieve accurate results, the general suggestion is to couple the fast information exchange in the rotationally variant PSO algorithm.

ISAPSO algorithm could be developed in which is based on the results found by the empirical analyses. It was rather preferable to work with rotation invariance property and face the lacking of directional diverse, than facing a bias phenomenon with rotation variance property. The new velocity update equation embodied a rotation matrix that is activated when the swarm is in the repulsion phase. This conception deals with the low directional diverse, providing a small perturbation in the opposite directions of social and gradient components, which restores part of the directional diverse. The algorithm is strictly rotationally invariant when swarm is in the attraction phase and rotationally invariant in stochastic sense when swarm is in the repulsion phase. A consistent numerical simulation showed that ISAPSO algorithm has the potential to be among the best well-known rotationally invariant algorithm, as the statistical hypothesis tests indicated statistical significant improvements when compared with other related PSOs.

5.1 Future research

Some future researches derived from this thesis are described below.

- To develop a framework containing all the codes related to the PSO algorithms and statistical hypothesis tests developed in this thesis. The goal is to provide an easy-to-use software able to deal with different optimization problems.
- To deepen the theoretical studies about relation between the following properties: rotation (in)variance and separability of the objective function. Furthermore, a consistent relation between the condition number and the algorithm's performance is of interest.

- To develop an adaptive PSO algorithm. This PSO version must varying its main parameters (inertia weight, cognitive and social coefficients) according to the needs of the swarm, i.e., if the swarm is converging to some prominent area of the search space, the parameters must represent this willing. The same endeavor has to be taken when the swarm is trapped into a local minima.

- To idealize a Tabu list for continuous optimization, similar to the one presented in the Tabu search algorithm. This idealized structure must forbid particles to reevaluate areas already visited during the search process.

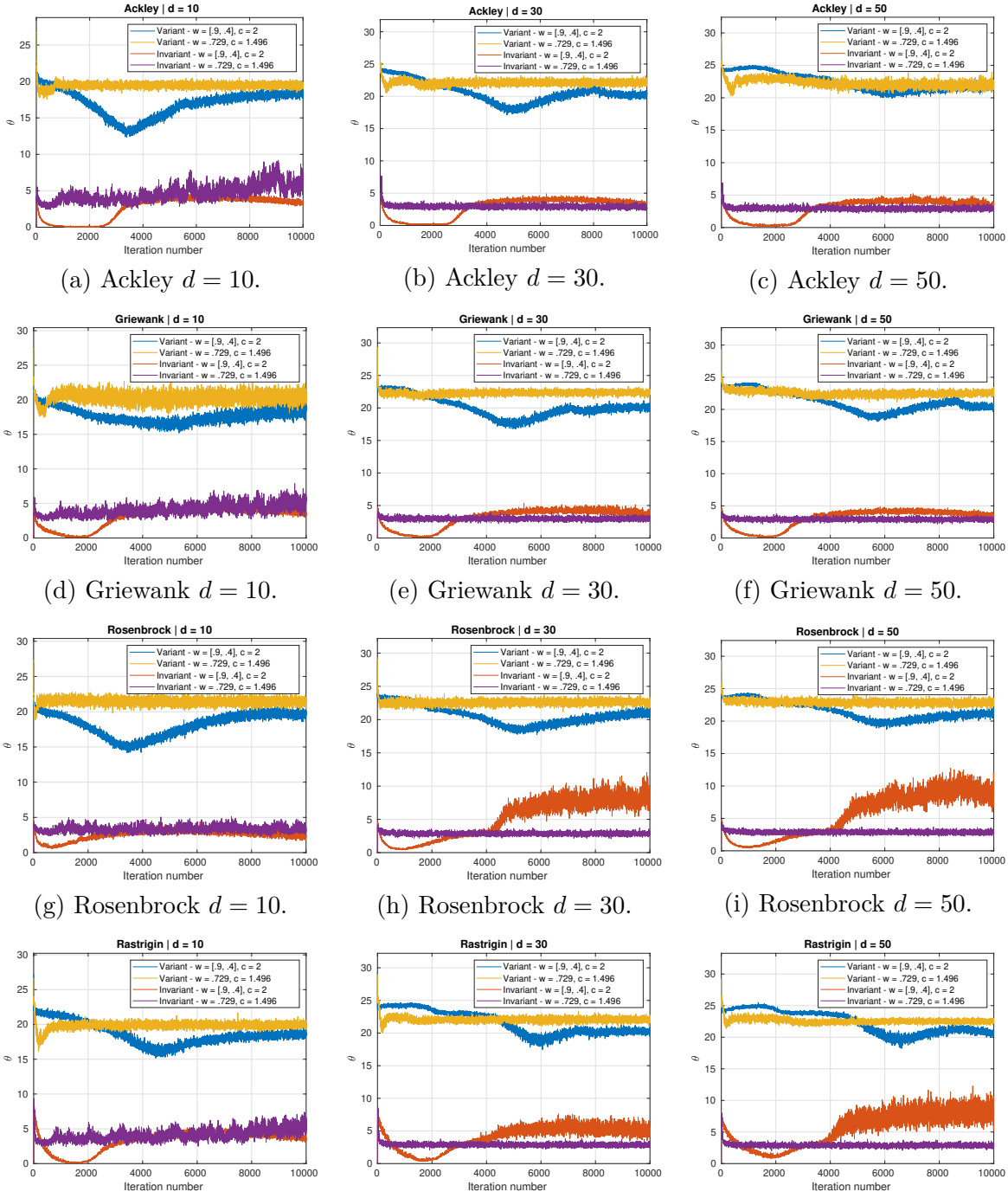
- To provide a rotationally variant PSO version free from bias regarding the directions of the coordinate axes.

- To hybridize swarms constituting with rotationally variant and rotationally invariant particles.

- To develop a parallel PSO version to accurate the best global position while the algorithm is still running. As soon as a new global best position is updated, a thread is triggered to accurate the global memory. The thread can use a deterministic technique such as Gradient descent and Newton's algorithms to optimize that position as initial guess.

APPENDIX A – Average angle analysis

Figure 19 – Average θ of the swarm according to the movement of particles along the iterations in high dimensions.



(a) Ackley $d = 10$. (b) Ackley $d = 30$. (c) Ackley $d = 50$.

(d) Griewank $d = 10$. (e) Griewank $d = 30$. (f) Griewank $d = 50$.

(g) Rosenbrock $d = 10$. (h) Rosenbrock $d = 30$. (i) Rosenbrock $d = 50$.

(j) Rastrigin $d = 10$. (k) Rastrigin $d = 30$. (l) Rastrigin $d = 50$.

Source: produced by the author.

APPENDIX B – PSO parameters

Table 27 – PSO parameters on De Jong's benchmark optimization problems.

<table>
<tr><th>f_i</th><th>SAPSO</th><th>GPSO</th><th>ARPSO</th><th>DGHPSOGS
(attraction phase)</th><th>DGHPSOGS
(repulsion phase)</th></tr>
<tr><td>f_1</td><td>$c_1 = 2$
$c_2 = 10^{-2}$</td><td rowspan="10">$c_1 = 2$
$c_2 = 2$</td><td rowspan="10">$w = 0.7 \rightarrow 0.4$
$c_1 = 2$
$c_2 = 2$</td><td>$w = 0.9 \rightarrow 0.4$
$c_1 = 2$
$c_2 = 2.1$</td><td rowspan="10">$w = 0.9 \rightarrow 0.4$
$c_1 = 2.1$
$c_2 = 2.1$</td></tr>
<tr><td>f_2</td><td>$c_1 = 2$
$c_2 = 10^{-3}$</td><td>$w = 0.9 \rightarrow 0.4$
$c_1 = 2.3$
$c_2 = 1.4$</td></tr>
<tr><td>f_3</td><td>$c_1 = 3$
$c_2 = 10^{-3}$</td><td>$w = 0.9 \rightarrow 0.4$
$c_1 = 2.1$
$c_2 = 2.8$</td></tr>
<tr><td>f_4</td><td>$c_1 = 3$
$c_2 = 10^{-2}$</td><td>$w = 0.9 \rightarrow 0.4$
$c_1 = 2.5$
$c_2 = 1.4$</td></tr>
<tr><td>f_5</td><td>$c_1 = 4$
$c_2 = 10^{-1}$</td><td>$w = 0.9 \rightarrow 0.4$
$c_1 = 2$
$c_2 = 2$</td></tr>
<tr><td>f_6</td><td rowspan="2">$c_1 = 2$
$c_2 = 10^{-2}$</td><td>$w = 0.9 \rightarrow 0.4$
$c_1 = 2.1$
$c_2 = 1.8$</td></tr>
<tr><td>f_7</td><td></td></tr>
<tr><td>f_8</td><td>$c_1 = 4$
$c_2 = 10^{-1}$</td><td>$w = 0.9 \rightarrow 0.4$
$c_1 = 2$
$c_2 = 2$</td></tr>
<tr><td>f_9</td><td rowspan="2">$c_1 = 2$
$c_2 = 10^{-2}$</td><td rowspan="2"></td></tr>
<tr><td>f_{10}</td></tr>
</table>

Table 28 – PSO parameters on CEC 2017 benchmark optimization problems.

f_i	SAPSO	GPSO	ARPSO	DGHPSOGS	SPSO	WPSO	BPSO	ISAPSO
f_1								
f_2				attraction phase:				
f_3				$w = 0.9 \rightarrow 0.4$				
f_4	$w = 0.7298$		$w = 0.7 \rightarrow 0.4$	$c_1 = 2$	$w = 0.721$	$w = 0.5$	$w = 0.7298$	$w = 0.7298$
f_5	$c_1 = 1.4962$	$c_1 = 2$	$c_1 = 2$	$c_2 = 10^{-2}$	$c_1 = 1.193$	$c_1 = 2$	$c_1 = 1.4962$	$c_1 = 1.4962$
f_6	$c_2 = 10^{-2}$	$c_2 = 2$	$c_2 = 2$	repulsion phase:	$c_2 = 1.193$	$c_2 = 2$	$c_2 = 1.4962$	$c_2 = 10^{-2}$
f_8				$w = 0.9 \rightarrow 0.4$				
f_9				$c_1 = 2.1$				
f_{10}				$c_2 = 2.1$				

Publication

Papers related to the thesis

- **Reginaldo Santos**, Gilvan Borges, Adam Santos, Moisés Silva, Claudomiro Sales, João C. W. A. Costa. *"Empirical study on rotation and information exchange in particle swarm optimization"*. Journal of Swarm and Evolutionary Computation, 2019. (status: under review)
- **Reginaldo Santos**, Gilvan Borges, Adam Santos, Moisés Silva, Claudomiro Sales, João C. W. A. Costa. *"A semi-autonomous particle swarm optimizer based on gradient information and diversity control for global optimization"*. Journal of Applied Soft Computing, 2018.

Papers submitted

- Caio Flexa, **Reginaldo Santos**, Walisson Gomes, Claudomiro Sales, João Costa. *"Mutual Equidistant-scattering Criterion: a new index for crisp clustering"*. Expert Systems With Applications, 2018.
- Walisson Gomes, **Reginaldo Santos**, Claudomiro Sales. *"Bachome: An Aid Tool in Natural Selection Teaching controlled by Genetic Algorithm"*. Revista do IEEE América Latina, 2017.

Papers published during the thesis period

- Walisson Gomes, **Reginaldo Santos**, Claudomiro Sales. *"An Improved Artificial Bee Colony Algorithm with Diversity Control"*. 7th Brazilian Conference on Intelligent Systems (BRACIS), 2018.
- Caio Flexa, **Reginaldo Santos**, Walissom Gomes, Claudomiro Sales. *"A Novel Equidistant-Scattering-based Cluster Index"*. 7th Brazilian Conference on Intelligent Systems (BRACIS), 2018.
- Moisés Silva, Adam Santos, **Reginaldo Santos**, Eloi Figueiredo, Claudomiro Sales, João C. W. A. Costa. *"Deep principal component analysis: An enhanced approach for structural damage identification"*. Structural Health Monitoring, 2018.

- Adam Santos, Moisés Silva, **Reginaldo Santos**, Eloi Figueiredo, Nuno Maia, João Costa. *" Applicability of an output-only structural damage detection based on transmissibility measurements and kernel principal component analysis"*. Simpósio Brasileiro de Telecomunicações e Processamento de Sinais, 2018.

- Moisés Silva, Adam Santos, **Reginaldo Santos**, Eloi Figueiredo, Claudomiro Sales, João C. W. A. Costa. *"Composing robust damage-sensitive features with deep neural networks"*. European Workshop on Structural Health Monitoring Series, 2018.

- Moisés Silva, Adam Santos, **Reginaldo Santos**, Eloi Figueiredo, Claudomiro Sales, João C. W. A. Costa. *"Agglomerative concentric hypersphere clustering applied to structural damage detection"*. Mechanical Systems and Signal Processing, 2017.

- Adam Santos, **Reginaldo Santos**, Moisés Silva, Eloi Figueiredo, Claudomiro Sales, João C. W. A. Costa. *"A Global Expectation-Maximization Approach Based on Memetic Algorithm for Vibration-Based Structural Damage Detection"*. IEEE Transactions on Instrumentation and Measurement, 2017.

- Moisés Silva, Adam Santos, **Reginaldo Santos**, Eloi Figueiredo, Claudomiro Sales, João C. W. A. Costa. *"A novel unsupervised approach based on a genetic algorithm for structural damage detection in bridges"*. Engineering Applications of Artificial Intelligence, 2016.

- Adam Santos, Eloi Figueiredo, Moisés Silva, **Reginaldo Santos**, Claudomiro Sales, João C. W. A. Costa. *"Genetic-based EM algorithm to improve the robustness of Gaussian mixture models for damage detection in bridges"*. Structural Control & Health Monitoring, 2016.

- Adam Santos, Moisés Silva, **Reginaldo Santos**, Eloi Figueiredo, Claudomiro Sales, João C. W. A. Costa. *"A global expectation-maximization based on memetic swarm optimization for structural damage detection"*. Structural Health Monitoring, 2016.

- Manoel Afonso Lima, Claudomiro Sales, Adam Santos, **Reginaldo Santos**, Moisés Silva, João C. W. A. Costa, Marissa Carvalho, Michel Cruz. *"A framework for data compression and damage detection in structural health monitoring applied on a laboratory three-story structure. Revista Brasileira de Computação Aplicada"*. Revista Brasileira de Computação Aplicada, 2016.

Bibliography

ACHARYA, J.; MEHTA, M.; SAINI, B. Particle swarm optimization based load balancing in cloud computing. In: *2016 International Conference on Communication and Electronics Systems (ICCES)*. [S.l.: s.n.], 2016. p. 1–4. Cited on page 29.

AHMED, A. et al. Particle swarm optimization for n-queens problem. v. 1, 05 2012. Cited on page 23.

AKSU, I. O.; COBAN, R. Training the multifeedback-layer neural network using the particle swarm optimization algorithm. In: *2013 International Conference on Electronics, Computer and Computation (ICECCO)*. [S.l.: s.n.], 2013. p. 172–175. Cited on page 23.

ARUMUGAM, M. S.; RAO, M. V. C. On the performance of the particle swarm optimization algorithm with various inertia weight variants for computing optimal control of a class of hybrid systems. *Discrete Dynamics in Nature and Society*, Hindawi Limited, v. 2006, p. 1–17, 2006. Cited 3 times on pages 48, 49, and 76.

AWAD, N. H. et al. *Problem Definitions and Evaluation Criteria for the CEC 2017 Special Session and Competition on Single Objective Real-Parameter Numerical Optimization*. [S.l.], 2017. Cited on page 37.

BANSAL, J. C. et al. Inertia weight strategies in particle swarm optimization. In: *2011 Third World Congress on Nature and Biologically Inspired Computing*. [S.l.: s.n.], 2011. p. 633–640. Cited on page 49.

BEHESHTI, Z.; SHAMSUDDIN, S. M. Non-parametric particle swarm optimization for global optimization. *Applied Soft Computing*, v. 28, p. 345 – 359, 2015. ISSN 1568-4946. Cited on page 26.

BEHESHTI, Z.; SHAMSUDDIN, S. M. H.; HASAN, S. Mpso: Median-oriented particle swarm optimization. *Applied Mathematics and Computation*, v. 219, n. 11, p. 5817 – 5836, 2013. ISSN 0096-3003. Cited on page 26.

BENTO, D. et al. Genetic algorithm and particle swarm optimization combined with powell method. *AIP Conference Proceedings*, v. 1558, n. 1, p. 578–581, 2013. Cited on page 23.

BERGH, F. van den; ENGELBRECHT, A. A study of particle swarm optimization particle trajectories. *Information Sciences*, v. 176, n. 8, p. 937 – 971, 2006. ISSN 0020-0255. Cited on page 24.

BIYANTO, T. R. et al. Killer whale algorithm: An algorithm inspired by the life of killer whale. *Procedia Computer Science*, v. 124, p. 151 – 157, 2017. ISSN 1877-0509. 4th Information Systems International Conference 2017, ISICO 2017, 6-8 November 2017, Bali, Indonesia. Cited on page 33.

BLUM, C. et al. Hybrid metaheuristics in combinatorial optimization: A survey. *Applied Soft Computing*, v. 11, n. 6, p. 4135 – 4151, 2011. ISSN 1568-4946. Cited on page 33.

BOCHENEK, B.; FORYŚ, P. Structural optimization for post-buckling behavior using particle swarms. *Structural and Multidisciplinary Optimization*, v. 32, n. 6, p. 521–531, Dec 2006. ISSN 1615-1488. Cited on page 46.

BONYADI, M. R.; MICHALEWICZ, Z. A locally convergent rotationally invariant particle swarm optimization algorithm. *Swarm Intelligence*, v. 8, n. 3, p. 159–198, Sep 2014. Cited 5 times on pages 32, 35, 36, 62, and 89.

BONYADI, M. R.; MICHALEWICZ, Z.; LI, X. An analysis of the velocity updating rule of the particle swarm optimization algorithm. *Journal of Heuristics*, v. 20, n. 4, p. 417–452, Aug 2014. Cited 4 times on pages 32, 36, 62, and 89.

BOUSSAïD, I.; LEPAGNOT, J.; SIARRY, P. A survey on optimization metaheuristics. *Information Sciences*, v. 237, p. 82 – 117, 2013. ISSN 0020-0255. Prediction, Control and Diagnosis using Advanced Neural Computations. Cited on page 33.

BRATTON, D.; KENNEDY, J. Defining a standard for particle swarm optimization. In: *2007 IEEE Swarm Intelligence Symposium*. [S.l.: s.n.], 2007. p. 120–127. Cited on page 46.

CAMCI, E. et al. An aerial robot for rice farm quality inspection with type-2 fuzzy neural networks tuned by particle swarm optimization-sliding mode control hybrid algorithm. *Swarm and Evolutionary Computation*, v. 41, p. 1 – 8, 2018. ISSN 2210-6502. Cited on page 29.

CH, S.; MATHUR, S. Particle swarm optimization trained neural network for aquifer parameter estimation. *KSCE Journal of Civil Engineering*, v. 16, n. 3, p. 298–307, Mar 2012. ISSN 1976-3808. Cited on page 23.

CHEN, K. et al. A hybrid particle swarm optimizer with sine cosine acceleration coefficients. *Information Sciences*, v. 422, p. 218 – 241, 2018. ISSN 0020-0255. Cited on page 26.

CHEN, Y. et al. Particle swarm optimizer with two differential mutation. *Applied Soft Computing*, v. 61, p. 314 – 330, 2017. ISSN 1568-4946. Cited on page 26.

CHEN, Y. et al. Particle swarm optimizer with crossover operation. *Engineering Applications of Artificial Intelligence*, v. 70, p. 159 – 169, 2018. ISSN 0952-1976. Cited on page 26.

CHENEY, W.; KINCAID, D. R. *Linear Algebra: Theory And Applications*. [S.l.]: Jones & Bartlett Learning, 2008. ISBN 0763750204. Cited on page 113.

CHOUIKHI, N. et al. Pso-based analysis of echo state network parameters for time series forecasting. *Applied Soft Computing*, v. 55, p. 211 – 225, 2017. ISSN 1568-4946. Cited on page 42.

CLERC, M. The swarm and the queen: towards a deterministic and adaptive particle swarm optimization. In: *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*. [S.l.: s.n.], 1999. v. 3, p. 1957 Vol. 3. Cited 2 times on pages 28 and 36.

CLERC, M. Confinements and Biases in Particle Swarm Optimisation. 9 pages. 2006. Cited on page 46.

CLERC, M. *Randomness matters*. [S.l.], 2012. 14 pages. Cited 4 times on pages 26, 34, 64, and 114.

CLERC, M. Standard particle swarm optimisation. 15 pages. 2012. Cited 6 times on pages 31, 35, 36, 61, 80, and 82.

CLERC, M.; KENNEDY, J. The particle swarm - explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation*, v. 6, n. 1, p. 58–73, Feb 2002. ISSN 1089-778X. Cited 7 times on pages 24, 29, 35, 36, 46, 49, and 73.

CUEVAS, E. et al. Circle detection using electro-magnetism optimization. *Information Sciences*, v. 182, n. 1, p. 40 – 55, 2012. ISSN 0020-0255. Nature-Inspired Collective Intelligence in Theory and Practice. Cited on page 33.

DERRAC, J. et al. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation*, v. 1, n. 1, p. 3 – 18, 2011. ISSN 2210-6502. Cited 3 times on pages 39, 122, and 124.

DORIGO, M.; MANIEZZO, V.; COLORNI, A. Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, v. 26, n. 1, p. 29–41, Feb 1996. ISSN 1083-4419. Cited on page 33.

DUAN, Y.; YING, S. A particle swarm optimization algorithm with ant search for solving traveling salesman problem. In: *2009 International Conference on Computational Intelligence and Security*. [S.l.: s.n.], 2009. v. 2, p. 137–141. Cited on page 23.

DURO, J. A.; OLIVEIRA, J. V. de. Particle swarm optimization applied to the chess game. In: *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*. [S.l.: s.n.], 2008. p. 3702–3709. ISSN 1089-778X. Cited on page 23.

DURRETT, R. *Probability: Theory and Examples*. [S.l.]: Cambridge University Press, 2010. (Cambridge Series in Statistical and Probabilistic Mathematics). ISBN 9781139491136. Cited on page 64.

EBERHART, R. C.; KENNEDY, J. A new optimizer using particle swarm theory. *Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, p. 39–43, 1995. Cited 6 times on pages 23, 26, 28, 36, 41, and 117.

EBERHART, R. C.; SHI, Y. Comparing inertia weights and constriction factors in particle swarm optimization. In: *Proceedings of the 2000 Congress on Evolutionary Computation. CEC00 (Cat. No.00TH8512)*. [S.l.: s.n.], 2000. v. 1, p. 84–88 vol.1. Cited 2 times on pages 28 and 36.

EBERHART, R. C.; SHI, Y. Tracking and optimizing dynamic systems with particle swarms. In: *Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No.01TH8546)*. [S.l.: s.n.], 2001. v. 1, p. 94–100 vol. 1. Cited on page 49.

ESPITIA, H. E.; SOFRONY, J. I. Statistical analysis for vortex particle swarm optimization. *Applied Soft Computing*, v. 67, p. 370 – 386, 2018. ISSN 1568-4946. Cited on page 26.

EUSUFF, M.; LANSEY, K.; PASHA, F. Shuffled frog-leaping algorithm: A memetic meta-heuristic for discrete optimization. *Engineering Optimization*, Taylor and Francis Ltd., v. 38, n. 2, p. 129–154, 3 2006. ISSN 0305-215X. Cited on page 33.

FENG, Y. et al. Chaotic inertia weight in particle swarm optimization. In: *Second International Conference on Innovative Computing, Informatio and Control (ICICIC 2007)*. [S.l.: s.n.], 2007. p. 475–475. Cited on page 49.

FLETCHER, R. *Practical Methods of Optimization; (2Nd Ed.)*. New York, NY, USA: Wiley-Interscience, 1987. ISBN 0-471-91547-5. Cited on page 53.

FRIEDMAN, M. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, Taylor & Francis, v. 32, n. 200, p. 675–701, 1937. Cited on page 122.

GBENGA, D. E.; RAMLAN, E. I. Understanding the limitations of particle swarm algorithm for dynamic optimization tasks: A survey towards the singularity of pso for swarm robotic applications. *ACM Comput. Surv.*, ACM, New York, NY, USA, v. 49, n. 1, p. 8:1–8:25, jul. 2016. ISSN 0360-0300. Cited on page 26.

GLOVER, F. Tabu search—part i. *ORSA Journal on Computing*, Institute for Operations Research and the Management Sciences (INFORMS), v. 1, n. 3, p. 190–206, aug 1989. Cited on page 24.

GLOVER, F. Tabu search—part II. *ORSA Journal on Computing*, Institute for Operations Research and the Management Sciences (INFORMS), v. 2, n. 1, p. 4–32, feb 1990. Cited on page 24.

GOLDBERG, D. E. *Genetic Algorithms in Search, Optimization and Machine Learning*. 1st. ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1989. ISBN 0201157675. Cited 2 times on pages 23 and 42.

GOU, J. et al. A novel improved particle swarm optimization algorithm based on individual difference evolution. *Applied Soft Computing*, v. 57, p. 468 – 481, 2017. ISSN 1568-4946. Cited on page 26.

GRIMALDI, E. A. et al. Pso as an effective learning algorithm for neural network applications. In: *Proceedings. ICCEA 2004. 2004 3rd International Conference on Computational Electromagnetics and Its Applications, 2004.* [S.l.: s.n.], 2004. p. 557–560. Cited on page 42.

GUDISE, V. G.; VENAYAGAMOORTHY, G. K. Comparison of particle swarm optimization and backpropagation as training algorithms for neural networks. In: *Swarm Intelligence Symposium, 2003. SIS '03. Proceedings of the 2003 IEEE*. [S.l.: s.n.], 2003. p. 110–117. Cited on page 42.

HAN, F.; LIU, Q. A diversity-guided hybrid particle swarm optimization based on gradient search. *Neurocomputing*, v. 137, p. 234 – 240, 2014. ISSN 0925-2312. Advanced Intelligent Computing Theories and MethodologiesSelected papers from the 2012 Eighth International Conference on Intelligent Computing (ICIC 2012). Cited 8 times on pages 25, 30, 34, 35, 36, 60, 99, and 101.

HAN, F.; LIU, Q. An improved hybrid pso based on arpso and the quasi-newton method. In: TAN, Y. et al. (Ed.). *Advances in Swarm and Computational Intelligence*. Cham: Springer International Publishing, 2015. p. 460–467. ISBN 978-3-319-20466-6. Cited on page 34.

HANSEN, E. Numerical methods for unconstrained optimization and nonlinear equations (j. e. dennis, jr., and robert b. schnabel). *SIAM Review*, Society for Industrial & Applied Mathematics (SIAM), v. 28, n. 3, p. 417–419, sep 1986. Cited on page 52.

HANSEN, N. et al. Impacts of invariance in search: When cma-es and pso face ill-conditioned and non-separable problems. *Applied Soft Computing*, v. 11, n. 8, p. 5755 – 5769, 2011. ISSN 1568-4946. Cited on page 54.

HAO, Z. F.; GUO, G. H.; HUANG, H. A particle swarm optimization algorithm with differential evolution. In: *2007 International Conference on Machine Learning and Cybernetics*. [S.l.: s.n.], 2007. v. 2, p. 1031–1035. ISSN 2160-133X. Cited on page 23.

HARIYA, Y.; SHINDO, T.; JIN'NO, K. A novel particle swarm optimization algorithm for non-separable and ill-conditioned problems. In: *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. [S.l.: s.n.], 2016. p. 002110–002115. Cited on page 89.

HART, W. E.; KRASNOGOR, N.; SMITH, J. E. Memetic evolutionary algorithms. In: _____. *Recent Advances in Memetic Algorithms*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005. p. 3–27. ISBN 978-3-540-32363-1. Cited on page 69.

HEPPNER, F.; GRENANDER, U. A stochastic nonlinear model for coordinated bird flocks. In: KRASNER, E. (Ed.). *The ubiquity of chaos*. [S.l.]: AAAS Publications, 1990. p. 233–238. Cited on page 41.

HEREFORD, J. M.; GERLACH, H. Integer-valued particle swarm optimization applied to sudoku puzzles. In: *2008 IEEE Swarm Intelligence Symposium*. [S.l.: s.n.], 2008. p. 1–7. Cited on page 23.

HOLLANDER, M.; WOLFE, D. A.; CHICKEN, E. *Nonparametric Statistical Methods*. [S.l.]: Wiley, 2013. ISBN 0470387378. Cited 3 times on pages 39, 122, and 125.

HORN, R. A.; JOHNSON, C. R. *Matrix Analysis*. 2nd. ed. New York, NY, USA: Cambridge University Press, 2012. ISBN 0521548233, 9780521548236. Cited on page 79.

HORST, R.; TUY, H. *Global Optimization: Deterministic Approaches*. [S.l.]: Springer Berlin Heidelberg, 1996. ISBN 9783540610380. Cited 3 times on pages 24, 51, and 66.

HOSSEINI, S.; KHALED, A. A. A survey on the imperialist competitive algorithm metaheuristic: Implementation in engineering domain and directions for future research. *Applied Soft Computing*, v. 24, p. 1078 – 1094, 2014. ISSN 1568-4946. Cited on page 33.

HU, X.; EBERHART, R. Solving constrained nonlinear optimization problems with particle swarm optimization. In: *6th World Multiconference on Systemics, Cybernetics and Informatics (SCI 2002*. [S.l.: s.n.], 2002. p. 203–206. Cited on page 46.

HUANG, X. et al. Research on particle swarm optimization and its industrial application. In: *Third International Conference on Natural Computation (ICNC 2007)*. [S.l.: s.n.], 2007. v. 3, p. 725–729. ISSN 2157-9555. Cited on page 23.

INSTITUTE, A. N. S.; ELECTRICAL, I. of; ENGINEERS, E. Ieee standard for binary floating-point arithmetic. *ANSI/IEEE Std 754-1985*, Mar 1985. Cited on page 65.

INSTITUTE, A. N. S.; ELECTRICAL, I. of; ENGINEERS, E. Ieee standard for floating-point arithmetic. *IEEE Std 754-2008*, Aug 2008. Cited on page 65.

ISSA, M. et al. Asca-pso: Adaptive sine cosine optimization algorithm integrated with particle swarm for pairwise local sequence alignment. *Expert Systems with Applications*, v. 99, p. 56 – 70, 2018. ISSN 0957-4174. Cited on page 26.

JANSON, S.; MIDDENDORF, M. On trajectories of particles in pso. In: *2007 IEEE Swarm Intelligence Symposium*. [S.l.: s.n.], 2007. p. 150–155. Cited 2 times on pages 24 and 64.

JENSI, R.; JIJI, G. W. An enhanced particle swarm optimization with levy flight for global optimization. *Applied Soft Computing*, v. 43, p. 248 – 261, 2016. ISSN 1568-4946. Cited on page 26.

JONG, K. A. D. *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*. Tese (Doutorado) — University of Michigan, Ann Arbor, MI, USA, 1975. AAI7609381. Cited 2 times on pages 37 and 98.

JR., I. F. et al. A brief review of nature-inspired algorithms for optimization. *CoRR*, abs/1307.4186, 2013. Cited on page 33.

JUANG, C.-F. A hybrid of genetic algorithm and particle swarm optimization for recurrent network design. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, v. 34, n. 2, p. 997–1006, April 2004. ISSN 1083-4419. Cited on page 23.

KARABOGA, D. *An idea based on honey bee swarm for numerical optimization*. [S.l.], 2005. Cited on page 33.

KENNEDY, J. The particle swarm: social adaptation of knowledge. In: *Evolutionary Computation, 1997., IEEE International Conference on*. [S.l.: s.n.], 1997. p. 303–308. Cited 5 times on pages 23, 26, 50, 73, and 88.

KENNEDY, J. The particle swarm: social adaptation of knowledge. In: *Evolutionary Computation, 1997., IEEE International Conference on*. [S.l.: s.n.], 1997. p. 303–308. Cited 2 times on pages 28 and 36.

KENNEDY, J. *Personal Communication with Dr. W. Spears*. 2007. Cited on page 81.

KENNEDY, J.; EBERHART, R. C. Particle swarm optimization. In: *Proceedings of the IEEE International Conference on Neural Networks*. [S.l.: s.n.], 1995. p. 1942–1948. Cited 7 times on pages 23, 26, 28, 36, 41, 42, and 117.

KHALIFA, M. H. et al. Particle swarm optimization for deep learning of convolution neural network. In: *2017 Sudan Conference on Computer Science and Information Technology (SCCSIT)*. [S.l.: s.n.], 2017. p. 1–5. Cited on page 29.

KHAN, A.; NIEMANN-DELIUS, C. Application of particle swarm optimization to the open pit mine scheduling problem. In: NIEMANN-DELIUS, C. (Ed.). *Proceedings of the 12th International Symposium Continuous Surface Mining - Aachen 2014*. Cham: Springer International Publishing, 2015. p. 195–212. ISBN 978-3-319-12301-1. Cited on page 23.

KRISHNANAND, K. N.; GHOSE, D. Detection of multiple source locations using a glowworm metaphor with applications to collective robotics. In: *Proceedings 2005 IEEE Swarm Intelligence Symposium, 2005. SIS 2005.* [S.l.: s.n.], 2005. p. 84–91. Cited on page 33.

KUMARI, R. V. S. L. et al. Optimal sizing of distributed generation using particle swarm optimization. In: *2017 International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICICT).* [S.l.: s.n.], 2017. p. 499–505. Cited on page 26.

LI, S.-F.; CHENG, C.-Y. Particle swarm optimization with fitness adjustment parameters. *Computers & Industrial Engineering*, v. 113, p. 831 – 841, 2017. ISSN 0360-8352. Cited on page 26.

LIANG, J. J. et al. Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. *IEEE Transactions on Evolutionary Computation*, v. 10, n. 3, p. 281–295, June 2006. ISSN 1089-778X. Cited on page 26.

LIU, Q.; HAN, F. A hybrid attractive and repulsive particle swarm optimization based on gradient search. In: HUANG, D.-S. et al. (Ed.). *Intelligent Computing Theories and Technology.* Berlin, Heidelberg: Springer Berlin Heidelberg, 2013. p. 155–162. ISBN 978-3-642-39482-9. Cited on page 34.

LOCATELLI, M. A note on the griewank test function. *Journal of Global Optimization*, v. 25, n. 2, p. 169–174, 2003. ISSN 1573-2916. Cited on page 103.

LU, Y. et al. Improved particle swarm optimization algorithm and its application in text feature selection. *Applied Soft Computing*, v. 35, p. 629 – 636, 2015. ISSN 1568-4946. Cited on page 42.

LYNN, N.; ALI, M. Z.; SUGANTHAN, P. N. Population topologies for particle swarm optimization and differential evolution. *Swarm and Evolutionary Computation*, v. 39, p. 24 – 35, 2018. ISSN 2210-6502. Cited on page 26.

LYNN, N.; SUGANTHAN, P. N. Ensemble particle swarm optimizer. *Applied Soft Computing*, v. 55, p. 533 – 548, 2017. ISSN 1568-4946. Cited on page 26.

MARINI, F.; WALCZAK, B. Particle swarm optimization (pso). a tutorial. *Chemometrics and Intelligent Laboratory Systems*, v. 149, Part B, p. 153 – 165, 2015. ISSN 0169-7439. Cited on page 26.

MATSUMOTO, M.; NISHIMURA, T. Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Trans. Model. Comput. Simul.*, ACM, New York, NY, USA, v. 8, n. 1, p. 3–30, jan. 1998. ISSN 1049-3301. Cited on page 65.

MOLER, C.; LOAN, C. V. Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later. *SIAM Review*, Society for Industrial & Applied Mathematics (SIAM), v. 45, n. 1, p. 3–49, jan 2003. Cited on page 62.

MONSON, C. K.; SEPPI, K. D. Exposing origin-seeking bias in pso. In: *Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation.* New York, NY, USA: ACM, 2005. (GECCO '05), p. 241–248. ISBN 1-59593-010-8. Cited on page 61.

MORAGLIO, A.; TOGELIUS, J. Geometric particle swarm optimization for the sudoku puzzle. In: *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation*. New York, NY, USA: ACM, 2007. (GECCO '07), p. 118–125. ISBN 978-1-59593-697-4. Cited on page 23.

MOSCATO, P. *On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms*. Pasadena, CA, 1989. Cited on page 69.

NAKAGAWA, N.; ISHIGAME, A.; YASUDA, K. Particle swarm optimization with approximate gradient. *IEEJ Transactions on Electrical and Electronic Engineering*, Wiley Subscription Services, Inc., A Wiley Company, v. 3, n. 5, p. 590–592, 2008. ISSN 1931-4981. Cited on page 49.

NOEL, M. M. A new gradient based particle swarm optimization algorithm for accurate computation of global minimum. *Applied Soft Computing*, v. 12, n. 1, p. 353 – 359, 2012. ISSN 1568-4946. Cited 8 times on pages 25, 26, 30, 34, 36, 59, 99, and 101.

NOEL, M. M.; JANNETT, T. C. Simulation of a new hybrid particle swarm optimization algorithm. In: *Thirty-Sixth Southeastern Symposium on System Theory, 2004. Proceedings of the*. [S.l.: s.n.], 2004. p. 150–153. ISSN 0094-2898. Cited 2 times on pages 34 and 101.

OKULEWICZ, M.; MAńDZIUK, J. The impact of particular components of the pso-based algorithm solving the dynamic vehicle routing problem. *Applied Soft Computing*, v. 58, p. 586 – 604, 2017. ISSN 1568-4946. Cited on page 42.

OZCAN, E. et al. Particle swarm optimization: Surfing the waves. In: *Proceedings of the Congress on Evolutionary Computation*. [S.l.]: IEEE Press, 1999. p. 6–9. Cited 2 times on pages 23 and 35.

OZCAN, E.; MOHAN, C. Analysis of a simple particle swarm optimization system. In: _____. *Intelligent Engineering Systems Through Artificial Neural Networks*. [S.l.: s.n.], 1998. v. 1998, p. 253–258. Cited on page 23.

POLI, R. Mean and variance of the sampling distribution of particle swarm optimizers during stagnation. *IEEE Transactions on Evolutionary Computation*, v. 13, n. 4, p. 712–721, Aug 2009. ISSN 1089-778X. Cited on page 24.

QIAN, X. L. L. J. S. X. An optimizing method based on autonomous animats: Fish-swarm algorithm. *Systems Engineering - Theory and Practice*, Systems Engineering - Theory and Practice, v. 22, n. 11, p. 32, 2002. Cited on page 33.

RABANAL, P.; RODRÍGUEZ, I.; RUBIO, F. Using river formation dynamics to design heuristic algorithms. In: AKL, S. G. et al. (Ed.). *Unconventional Computation*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007. p. 163–177. ISBN 978-3-540-73554-0. Cited on page 33.

RASHEDI, E.; NEZAMABADI-POUR, H.; SARYAZDI, S. Gsa: A gravitational search algorithm. *Inf. Sci.*, Elsevier Science Inc., New York, NY, USA, v. 179, n. 13, p. 2232–2248, jun. 2009. ISSN 0020-0255. Cited on page 33.

RATNAWEERA, A.; HALGAMUGE, S. K.; WATSON, H. C. Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. *IEEE Transactions on Evolutionary Computation*, v. 8, n. 3, p. 240–255, June 2004. ISSN 1089-778X. Cited 5 times on pages 28, 35, 48, 76, and 117.

RAUNIYAR, A.; ENGELSTAD, P.; MOEN, J. A new distributed localization algorithm using social learning based particle swarm optimization for internet of things. In: *2018 IEEE 87th Vehicular Technology Conference (VTC Spring)*. [S.l.: s.n.], 2018. p. 1–7. ISSN 2577-2465. Cited on page 29.

REYNOLDS, C. W. Flocks, herds and schools: A distributed behavioral model. In: *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*. New York, NY, USA: ACM, 1987. (SIGGRAPH '87), p. 25–34. ISBN 0-89791-227-6. Cited on page 41.

REYNOLDS, C. W. Flocks, herds and schools: A distributed behavioral model. *SIGGRAPH Comput. Graph.*, ACM, New York, NY, USA, v. 21, n. 4, p. 25–34, ago. 1987. ISSN 0097-8930. Cited on page 41.

ROBATI, A. et al. Balanced fuzzy particle swarm optimization. *Applied Mathematical Modelling*, v. 36, n. 5, p. 2169 – 2177, 2012. ISSN 0307-904X. Cited on page 26.

SAAD, N. H.; EL-SATTAR, A. A.; MANSOUR, A. E.-A. M. A novel control strategy for grid connected hybrid renewable energy systems using improved particle swarm optimization. *Ain Shams Engineering Journal*, v. 9, n. 4, p. 2195 – 2214, 2018. ISSN 2090-4479. Cited on page 29.

SADOLLAH, A. et al. Mine blast algorithm: A new population based algorithm for solving constrained engineering optimization problems. *Applied Soft Computing*, v. 13, n. 5, p. 2592 – 2612, 2013. ISSN 1568-4946. Cited on page 33.

SALOMON, R. Re-evaluating genetic algorithm performance under coordinate rotation of benchmark functions. a survey of some theoretical and practical aspects of genetic algorithms. *Biosystems*, v. 39, n. 3, p. 263 – 278, 1996. ISSN 0303-2647. Cited on page 64.

SANTOS, A. et al. A global expectation-maximization based on memetic swarm optimization for structural damage detection. *Structural Health Monitoring*, v. 15, n. 5, p. 610–625, 2016. Cited on page 23.

SANTOS, R. et al. A semi-autonomous particle swarm optimizer based on gradient information and diversity control for global optimization. *Applied Soft Computing*, v. 69, p. 330 – 343, 2018. ISSN 1568-4946. Cited 2 times on pages 36 and 69.

SCHMITT, M.; WANKA, R. Particles prefer walking along the axes. In: *Proceeding of the fifteenth annual conference companion on Genetic and evolutionary computation conference companion - GECCO '13 Companion*. [S.l.]: ACM Press, 2013. Cited on page 24.

SCHMITT, M.; WANKA, R. Particles prefer walking along the axes: Experimental insights into the behavior of a particle swarm. *CoRR*, abs/1303.6145, 2013. Cited on page 24.

SHAMI, T. M.; GRACE, D.; BURR, A. Load balancing and control using particle swarm optimisation in 5g heterogeneous networks. In: *2018 European Conference on Networks and Communications (EuCNC)*. [S.l.: s.n.], 2018. p. 1–9. ISSN 2575-4912. Cited on page 29.

SHEHAB, M.; KHADER, A. T.; AL-BETAR, M. A. A survey on applications and variants of the cuckoo search algorithm. *Applied Soft Computing*, 2017. ISSN 1568-4946. Cited on page 33.

SHI, Y.; EBERHART, R. A modified particle swarm optimizer. In: *1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98TH8360)*. [S.l.: s.n.], 1998. p. 69–73. Cited 9 times on pages 26, 28, 36, 47, 48, 60, 76, 79, and 117.

SHI, Y.; EBERHART, R. C. Parameter selection in particle swarm optimization. In: PORTO, V. W. et al. (Ed.). *Evolutionary Programming VII*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998. p. 591–600. ISBN 978-3-540-68515-9. Cited 3 times on pages 26, 48, and 49.

SHI, Y.; EBERHART, R. C. Empirical study of particle swarm optimization. In: *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*. [S.l.: s.n.], 1999. v. 3, p. 1950 Vol. 3. Cited 7 times on pages 26, 28, 35, 36, 48, 49, and 117.

SNYMAN, J. A. *Practical mathematical optimization : an introduction to basic optimization theory and classical and new gradient-based algorithms*. [S.l.]: Springer-Verlag, 2005. Cited on page 55.

SÖRENSEN, K. Metaheuristics—the metaphor exposed. *International Transactions in Operational Research*, v. 22, n. 1, p. 3–18, 2015. ISSN 1475-3995. Cited on page 33.

SPEARS, W. M.; GREEN, D. T.; SPEARS, D. F. Biases in particle swarm optimization. *Int. J. Swarm. Intell. Res.*, IGI Global, Hershey, PA, USA, v. 1, n. 2, p. 34–57, abr. 2010. ISSN 1947-9263. Cited 6 times on pages 24, 31, 61, 64, 81, and 89.

SUGANTHAN, P. N. Particle swarm optimiser with neighbourhood operator. In: *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*. [S.l.: s.n.], 1999. v. 3, p. 1962 Vol. 3. Cited 2 times on pages 48 and 49.

TEWOLDE, G. S.; HANNA, D. M.; HASKELL, R. E. Enhancing performance of pso with automatic parameter tuning technique. In: *2009 IEEE Swarm Intelligence Symposium*. [S.l.: s.n.], 2009. p. 67–73. Cited on page 26.

THABIT, S.; MOHADES, A. Multi-robot path planning based on multi-objective particle swarm optimization. *IEEE Access*, v. 7, p. 2138–2147, Jan 2019. ISSN 2169-3536. Cited on page 29.

TIAN, D.; SHI, Z. Mpso: Modified particle swarm optimization and its applications. *Swarm and Evolutionary Computation*, 2018. ISSN 2210-6502. Cited on page 26.

VESTERSTRØM, J.; RIGET, J. A diversity-guided particle swarm optimizer - the arpso. *EVALife Technical Report*, n. 2002-02, 2002. Cited 7 times on pages 25, 30, 36, 58, 75, 99, and 101.

VILOVIĆ, I.; BURUM, N.; MILIĆ, D. Using particle swarm optimization in training neural network for indoor field strength prediction. In: *2009 International Symposium ELMAR*. [S.l.: s.n.], 2009. p. 275–278. ISSN 1334-2630. Cited on page 23.

WANG, Y. R.; LIN, H. L.; YANG, L. Swarm refinement pso for solving n-queens problem. In: *2012 Third International Conference on Innovations in Bio-Inspired Computing and Applications*. [S.l.: s.n.], 2012. p. 29–33. Cited on page 23.

WEDYAN, A.; WHALLEY, J.; NARAYANAN, A. Hydrological cycle algorithm for continuous optimization problems. *Journal of Optimization*, Hindawi Limited, v. 2017, p. 1–25, 2017. Cited on page 33.

WILKE, D. N.; KOK, S.; GROENWOLD, A. A. Comparison of linear and classical velocity update rules in particle swarm optimization: notes on diversity. *International Journal for Numerical Methods in Engineering*, v. 70, n. 8, p. 962–984, 2007. Cited 5 times on pages 26, 31, 62, 63, and 89.

WILKE, D. N.; KOK, S.; GROENWOLD, A. A. Comparison of linear and classical velocity update rules in particle swarm optimization: notes on scale and frame invariance. *International Journal for Numerical Methods in Engineering*, v. 70, n. 8, p. 985–1008, 2007. Cited 11 times on pages 26, 31, 32, 35, 36, 53, 55, 62, 63, 89, and 127.

WILSON, E. *Sociobiology: The New Synthesis*. [S.l.]: Belknap Press of Harvard University Press, 1975. ISBN 9780674816213. Cited on page 41.

WILSON, E. *Sociobiology: The New Synthesis*. [S.l.]: Belknap Press of Harvard University Press, 2000. ISBN 9780674000896. Cited on page 41.

WOLPERT, D. H.; MACREADY, W. G. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, v. 1, n. 1, p. 67–82, Apr 1997. ISSN 1089-778X. Cited on page 109.

XU, G.; YU, G. On convergence analysis of particle swarm optimization algorithm. *Journal of Computational and Applied Mathematics*, v. 333, p. 65 – 73, 2018. ISSN 0377-0427. Cited on page 26.

XU, S.-H. et al. A combination of genetic algorithm and particle swarm optimization for vehicle routing problem with time windows. *Sensors*, MDPI AG, v. 15, n. 9, p. 21033–21053, aug 2015. Cited on page 23.

YAMAGUCHI, T.; YASUDA, K. Adaptive particle swarm optimization; self-coordinating mechanism with updating information. In: *2006 IEEE International Conference on Systems, Man and Cybernetics*. [S.l.: s.n.], 2006. v. 3, p. 2303–2308. ISSN 1062-922X. Cited 2 times on pages 28 and 36.

YANG, X.-S. *Nature-Inspired Metaheuristic Algorithms*. [S.l.]: Luniver Press, 2008. ISBN 1905986106, 9781905986101. Cited on page 33.

YANG, X.-S. Firefly algorithms for multimodal optimization. In: WATANABE, O.; ZEUGMANN, T. (Ed.). *Stochastic Algorithms: Foundations and Applications*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009. p. 169–178. ISBN 978-3-642-04944-6. Cited on page 33.

YANG, X.-S. Flower pollination algorithm for global optimization. In: DURAND-LOSE, J.; JONOSKA, N. (Ed.). *Unconventional Computation and Natural Computation*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. p. 240–249. ISBN 978-3-642-32894-7. Cited on page 33.

YANG, X.-S.; HE, X. Bat algorithm: Literature review and applications. *Int. J. Bio-Inspired Comput.*, Inderscience Publishers, Inderscience Publishers, Geneva, SWITZERLAND, v. 5, n. 3, p. 141–149, jul. 2013. ISSN 1758-0366. Cited on page 33.

YU, J. J.; LI, V. O. A social spider algorithm for global optimization. *Applied Soft Computing*, v. 30, p. 614 – 627, 2015. ISSN 1568-4946. Cited on page 33.

YU, K.; WANG, X.; WANG, Z. Multiple learning particle swarm optimization with space transformation perturbation and its application in ethylene cracking furnace optimization. *Knowledge-Based Systems*, v. 96, p. 156 – 170, 2016. ISSN 0950-7051. Cited on page 26.

ZAHARA, E.; KAO, Y. T.; LIU, C. H. Gradient enhanced particle swarm optimization for unconstrained problems. In: *2009 Fourth International Conference on Innovative Computing, Information and Control (ICICIC)*. [S.l.: s.n.], 2009. p. 893–896. Cited on page 34.

ZAHARA, E.; KAO, Y. T.; SU, J. R. Enhancing particle swarm optimization with gradient information. In: *2009 Fifth International Conference on Natural Computation*. [S.l.: s.n.], 2009. v. 3, p. 251–254. ISSN 2157-9555. Cited on page 34.

ZAMBRANO-BIGIARINI, M.; CLERC, M.; ROJAS-MUJICA, R. Standard particle swarm optimisation 2011 at CEC-2013: A baseline for future PSO improvements. In: *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2013, Cancun, Mexico, June 20-23, 2013*. [S.l.: s.n.], 2013. p. 2337–2344. Cited on page 61.

ZAMBRANO-BIGIARINI, M.; CLERC, M.; ROJAS, R. Standard particle swarm optimisation 2011 at cec-2013: A baseline for future pso improvements. In: *2013 IEEE Congress on Evolutionary Computation*. [S.l.: s.n.], 2013. p. 2337–2344. ISSN 1089-778X. Cited on page 32.

ZANDI, Z.; AFJEI, E.; SEDIGHIZADEH, M. Reactive power dispatch using big bang-big crunch optimization algorithm for voltage stability enhancement. In: *2012 IEEE International Conference on Power and Energy (PECon)*. [S.l.: s.n.], 2012. p. 239–244. Cited on page 33.

ZAR, J. H. *Biostatistical Analysis (5th Edition)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2007. ISBN 0131008463. Cited 3 times on pages 122, 123, and 125.

ZHAN, Z. h. et al. Adaptive control of acceleration coefficients for particle swarm optimization based on clustering analysis. In: *2007 IEEE Congress on Evolutionary Computation*. [S.l.: s.n.], 2007. p. 3276–3282. ISSN 1089-778X. Cited 2 times on pages 28 and 36.

ZHAN, Z. H. et al. Adaptive particle swarm optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, v. 39, n. 6, p. 1362–1381, Dec 2009. ISSN 1083-4419. Cited 2 times on pages 28 and 36.

ZHAN, Z. H. et al. Orthogonal learning particle swarm optimization. *IEEE Transactions on Evolutionary Computation*, v. 15, n. 6, p. 832–847, Dec 2011. ISSN 1089-778X. Cited on page 26.

ZHANG, Q. et al. Vector coevolving particle swarm optimization algorithm. *Information Sciences*, v. 394-395, p. 273 – 298, 2017. ISSN 0020-0255. Cited on page 26.