UNIVERSIDADE ESTADUAL DE CAMPINAS

Faculdade de Engenharia Elétrica e de Computação

Liu Yi Ling

Participatory Search Algorithms and Applications

Algoritmo de Busca Participativa e Aplicações

Campinas
2016

Liu Yi Ling

Participatory Search Algorithms and Applications

Algoritmo de Busca Participativa e Aplicações

Doctorate thesis presented to the School of Electrical and Computer Engineering in partial fulfillment of the requirements for the degree of Doctor in Electrical Engineering. Concentration area: Automation

Tese de doutorado apresentada à Faculdade de Engenharia Elétrica e de Computação como parte dos requisitos exigidos para a obtenção do título de Doutora em Engenharia Elétrica. Área de concentração: Automação.

Orientador: Prof. Dr. Fernando Antonio Campos Gomide

Este exemplar corresponde à versão final da tese defendida pela aluna, e orientada pelo Prof. Dr. Fernando Antonio Campos Gomide

---

Campinas
2016

Agência(s) de fomento e nº(s) de processo(s): Não se aplica.

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca da Área de Engenharia e Arquitetura
Luciana Pietrosanto Milla - CRB 8/8129

L74p

Liu, Yi Ling, 1980-
Participatory search learning algorithms and applications / Liu Yi Ling. – Campinas, SP : [s.n.], 2016.

Orientador: Fernando Antônio Campos Gomide.
Tese (doutorado) – Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Otimização. 2. Algoritmos evolutivos. 3. Sistemas fuzzy. I. Gomide, Fernando Antônio Campos,1951-. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Informações para Biblioteca Digital

Título em outro idioma: Algoritmo de busca participativa e aplicações
Palavras-chave em inglês:
Otimization
Evolutionary algorithms
Fuzzy systems
Área de concentração: Automação
Titulação: Doutora em Engenharia Elétrica
Banca examinadora:
Fernando Antônio Campos Gomide [Orientador]
Maury Meirelles Gouvêa Júnior
Jorge Luis Machado do Amaral
Fábio Luiz Usberti
Fernando José Von Zuben
Data de defesa: 27-10-2016
Programa de Pós-Graduação: Engenharia Elétrica

COMISSÃO JULGADORA - TESE DE DOUTORADO

Candidato: Liu Yi Ling RA: 089280
Data da Defesa: 27 de outubro de 2016

Título da Tese: Participatory Search Algorithms and Applications (Algoritmo de Busca Participativa e Aplicações)

Prof. Dr. Fernando Antonio Campos Gomide (Presidente, FEEC/UNICAMP)
Prof. Dr. Maury Meirelles Gouvêa Júnior (PUC-MG)
Prof. Dr. Jorge Luis Machado do Amaral (UERJ)
Prof. Dr. Fábio Luiz Usberti (IC/UNICAMP)
Prof. Dr. Fernando José Von Zuben (FEEC/UNICAMP)

A ata de defesa, com as respectivas assinaturas dos membros da Comissão Julgadora, encontra-se no processo de vida acadêmica do aluno.

Acknowledgement

I would like to thank all the people who contributed in some way to the work developed in this thesis. First and foremost, I thank my advisor Professor Fernando Antonio Campos Gomide who has offered his unreserved help and guidance that lead me to my thesis step by step. His words can always inspire me and bring me to a higher level of thinking. Without his kind and patient help, it would be much more difficult to finish this thesis.

I am grateful to Professor Alexandre Falcão and Professor Fernando Von Zuben for their contributions offered during my presentation in the qualifying exam, when they carefully indicated paths and made inspiring comments that helped me to improve this work.

I also want to thank my friends who encouraged and supported me during the graduate days. The simple phrase, "thank you", cannot present how much their friendship means to me.

Finally, I am very grateful to my family, who provided me a fruitful and peaceful environment, so that I could concentrate on my study. Although they may hardly understand what writing thesis is and what I research on, my family always supported the decision I made. I am so lucky to have them as my family.

Abstract

Search is one of the most useful procedures employed in numerous situations such as optimization, machine learning, information processing and retrieval. This work introduces participatory search, a population-based search algorithm based on the participatory learning paradigm. Participatory search is an algorithm in which search progresses forming pools of compatible individuals, keeping the one that is the most compatible with the current best individual in the current population, and introducing random individuals in each algorithm step. Recombination is a convex combination modulated by the compatibility between individuals while mutation is an instance of differential variation modulated by compatibility between selected and recombined individuals. The nature of the recombination and mutation operators are studied. Convergence analysis of the algorithm is pursued within the framework of random search theory.

The participatory search algorithm with arithmetical-like recombination is evaluated using ten benchmark real-valued optimization problems, and its performance is compared against population-based optimization algorithms representative of the current state of the art in the area. The participatory search algorithm arithmetical recombination is also evaluated using a suite of twenty eight benchmark functions of the evolutionary, real-valued optimization competition of the IEEE CEC 2013 (IEEE *Congress on Evolutionary Computation*) to compare its performance against the competition winners. Computational results suggest that the participatory search algorithm is as good as the winners.

An application concerning development of fuzzy rule-based models from actual data is given. The performance of the models produced by participatory search algorithms are compared with a state of the art genetic fuzzy system approach. Experimental results suggest that the participatory search algorithm with arithmetical-like recombination performs best.

Keywords: Population-Based Search, Participatory Learning, Compatibility-Based Optimization, Fuzzy Modeling.

Resumo

Busca é um dos procedimentos mais úteis em inúmeras aplicações, entre elas otimização, aprendizagem de máquina, processamento e recuperação da informação. Este trabalho sugere busca participativa, um algoritmo populacional baseado no paradigma de aprendizagem participativa. Busca participativa é um algoritmo em que a busca prossegue orientada pela compatibilidade entre indivíduos de uma população, sempre mantendo o melhor indivíduo nas populações posteriores e introduzindo indivíduos aleatoriamente em cada passo do algoritmo. A recombinação é uma combinação convexa modulada pela compatibilidade entre os indivíduos e a mutação é similar à variação diferencial modulada pela compatibilidade entre indivíduos selecionados e recombinados. A natureza dos operadores de recombinação e mutação são estudados. A convergência dos algoritmos também é estudada no âmbito da teoria de busca aleatória.

O algoritmo de busca participativa com recombinação aritmética é valiado utilizando dez problemas de otimização considerados como *benchmarks* na literatura e seu desempenho é comparado com algoritmos populacionais representativos do estado da arte atual. O algoritmo de busca participativa também é valiado utilizando um conjunto de vinte e oito funções de uma competição realizada como parte do IEEE CEC 2013 (IEEE *Congress on Evolutionary Computation*) e é comparado com os algoritmos vencedores desta competição. Resultados computacionais mostram que o algoritmo de busca participativa com recombinação aritmética compete igualmente com os vencedores.

Aplicações dos algoritmos de busca participativa foram estudadas no contexto de modelagem linguística de dados. Para isso, utilizaram dados reais disponíveis na literatura, dados estes de diferentes naturezas e dimensionalidades. O desempenho dos algoritmos de busca participativa foi avaliado e comparado com um dos mais representativos sistemas genético *fuzzy* disponíveis na literatura. Os resultados computacionais corroboram que o algoritmo de busca participativa com recombinação aritmética é competitivo, computacionalmente simples e eficiente.

Palavras-chave: Busca Baseada em População, Aprendizagem Participativa, Otimização Baseada em Compatibilidade, Modelagem de Sistemas Fuzzy.

List of Figures

List of Tables

List of Acronyms

ACO	Ant Colony Optimization
ABA	Data of Abalone Age
ANA	Data of Categorical Analysis
BBO	Biogeography-Based Optimization
CGA	Canonical Genetic Algorithm
DB	Data Base
DE	Differential Evolution
DPSA	Differential Participatory Search with Arithmetical Recombination
DPST	Differential Participatory Search with Selective Transfer
EC	Evolutionary Computation
ELE	Data of Electric Maintenance
ES	Evolutionary Strategy
FRBS	Fuzzy Rule-Based Systems
FSMOGFS	Fast and Scalable Multiobjective Genetic Fuzzy System
FSMOGFS^e	FSMOGFS including fast error estimation
FSMOGFS+TUN	FSMOGFS with tuning of MF parameters and rule selection by SPEA2
FSMOGFS^e+TUN^e	FSMOGFS+TUN including fast error estimation
GA	Genetic Algorithm
GFRBS	Genetic Fuzzy Rule-Based Systems
GFS	Genetic Fuzzy Systems
IEEE CEC	IEEE Conference on Evolutionary Computation
KB	Knowledge Base
KEEL	Knowledge Extraction based on Evolutionary Learning
KNN	k-Nearest Neighbors Method
MF	Membership Function
MPG6	Data of Auto MPG6
MSE	Mean-Squared Error
PBIL	Probability-Based Incremental Learning
PCBLX	Parent-Centric BLX operator
PL	Participatory Learning
PS	Participatory Search
PSA	Participatory Search Algorithm

PSAR	Participatory Search with Arithmetical Recombination
PSO	Particle Swarm Optimization
PSST	Participatory Search with Selective Transfer
RB	Rule Base
RL	Reinforcement Learning
r-CGA	Real-Coded Genetic Algorithm
SBSMs	Similarity-Based Surrogate Models
SGA	Stud Genetic Algorithm
SPEA2	Improving the Strength Pareto Evolutionary Algorithm
WM	Wang and Mendel Algorithm
WM(3)	Rule Base Produced by WM, 3 Linguistic Labels for Each Variable
WM(5)	Rule Base Produced by WM, 5 Linguistic Labels for Each Variable
WM(7)	Rule Base Produced by WM, 7 Linguistic Labels for Each Variable

List of Symbols

S^t	population at step t
$v(t)$	current knowledge at step t
$z(t)$	input information at step t
α	basic learning rate
ρ_t	compatibility degree at step t
a_t	arousal index at step t
β	parameter that controls the rate of arousal
$S^{'}$	mating pool
N	population size
n	fixed length strings
s	an individual of S
$s^{'}$	an individual of $S^{'}$
r	remaining individuals
s^*	current best individual
$best$	best individual
f	objective function
ρ^s	compatibility degrees between s and s^*
$\rho^{s^{'}}$	compatibility degrees between $s^{'}$ and s^*
$p_{selected}$	selected individual
ρ_r	compatibility degrees for recombination
p_r	offspring from recombination
ρ_m	compatibility degree for mutation
p_m	mutated individuals
$last(S^t)$	last individual in population S^t
P	probability
$\mathbb{R}^n$	real space of n dimension
μ	probability measures
$R_{\epsilon,M}$	optimality region
σ	essential infimum
L	Lebesgue measure
$\mathcal{L}$	Lebesgue measurable sets

D	function for conceptual algorithm condition
U	uniforme distribution
$\mathcal{B}$	Borel set
H_0	null hypothesis

Contents

Chapter 1

Introduction

The development of algorithms for problem solving has been a major subject in applied mathematics, computer science and engineering. Evolutionary computation provides a path to solve complex problems in many areas, especially in optimization and system modeling. They are robust, do not need too much specialization to specific classes of problems, and deliver good solutions within reasonable time [Eiben and Smith, 2015]. Evolutionary computation technique is an abstraction from the theory of biological evolution that is used to biological evolution that is used to create optimization procedures or create optimization procedures or methodologies, usually implemented on methodologies, usually implemented on computers, that are used to solve computers, that are used to solve problems.

One of the most fertile and practical evolutionary computation approaches is the genetic algorithm (GA). The basic idea of GA is to maintain a population of individuals that evolves using selection, recombination, and mutation operators working in sequence during several steps called generations. A fitness function determine the chance of an individual to survive. Higher fitness means higher chance of survival. Computationally speaking, a genetic algorithm is a stochastic search algorithm biased towards better solutions. The search technique aims at finding the fittest individual in a population of individuals. Individuals are candidate solutions of a target problem.

Differential evolution (DE) is another important component of evolutionary computation. Like GA, DE is a population-based stochastic search scheme whose purpose is to find the fittest individual. It is fairly a fast and reasonably robust optimization method [Storn and Price, 1997]. The basic idea of DE is to maintain a population of candidate solutions and to create new candidate solutions for each solution by using one constant member and two random members working in sequence during generations. DE is elitist, that is, it keeps the candidate solutions that achieves the best fitness function value in the population.

Learning is not a pure context-free process situated in a neutral environment. It is affected by the state of the learner. In particular, in the actual survival of the fittest saga there appears to be an additional process going on. Besides being determined by some external requirement, reproduction is always strongly affected by the population itself. It is well known that genetic compatibility is a mechanism by which individuals obtain fitness benefit through mate choice [Agbali et al., 2010]. Reproductive assurance and the benefits of self-compatibility may also be strongly influenced by density, diversity, or population size [Busch, 2005]. These observations suggest the possibility of building participatory search algorithms in which the population plays a role in determining reproduction and its

ensuing evolutionary path. In this sense, learning can be viewed as a sort of fitness learning once fitness shapes the reproductive suitability of individuals. When the population affects evolution, fitness of each individual results from the combination of its own objective and its compatibility with different individuals. In natural environments, often the effect of the population in crafting fitness is to try to make individuals more like themselves [Yager, 2000].

One way to address the role that the population itself plays in evolution is participatory learning [Yager, 1990]. The basic hypothesis of the participatory learning paradigm is that learning occurs in the framework of what is already learned or believed. The implication of this is that every aspect of the learning process is affected and guided by the current belief system. The name emphasizes the fact that in learning we are in an environment in which the current knowledge of what we are attempting to learn participates in the process of learning about itself. Participatory learning introduces compatibility between individuals to account for their involvement in the learning process. The notion of compatibility, likewise similarity, is recurrent in many machine and evolutionary learning schemes. For example, the ratio between the number of distinct genes and the total number of genes, called difference degree in [Chen and Wang, 2010], is a way to measure similarity of a pair of chromosomes [Chen and Yin, 2012]. The idea of difference degree together with unimodal normal distribution crossover and a variant of non-uniform mutation have shown to produce efficient real-coded genetic algorithms. Similarity has also been shown to be effective in evolutionary multiobjective optimization once similarity-based mating schemes have shown to increase diversity [Ishibuchi et al., 2007]. For instance, [Ishibuchi et al., 2008] introduces a similarity-based mating scheme in which an extreme solution is selected in the current population as a parent, and a similar solution to the parent selected is chosen as a mate. In addition to improve diversity, similarity-based mating schemes help to avoid recombination of dissimilar mates from which good offspring are unlikely to be produced. Other forms of participation in evolutionary learning include symmetric and asymmetric interactions between individuals. Crossover is a symmetric variation mechanism reflecting interaction through a recombination procedure that exchanges information from congruent parts of chromosomes. Selective transfer [Birchenhall et al., 1997] is an asymmetric variation mechanism that casts interaction via recombination [Liu and Gomide, 2013c], [Liu and Gomide, 2013a].

Participatory fuzzy systems are fuzzy systems with added participatory components. They result from the hybridization of participatory search algorithms and fuzzy systems within the structure of soft computing and computational intelligence. Participatory fuzzy systems offer a systematic approach and an effective design tool to address challenging issues such as fuzzy modeling, classification, and system control. Hybridization has been a fertile avenue for intelligent systems development. For instance, the combination of fuzzy systems with genetic learning induces genetic fuzzy systems (GFS) [Cordón et al., 2001], [Herrera, 2008]. Currently, there is a growing interest in data-driven fuzzy modeling using GFS. For instance, multiobjective evolutionary algorithms have been developed to build linguistic fuzzy rule-based systems with embedded genetic database learning for fast learning of parsimonious and accurate models [Alcalá et al., 2011], [Antonelli et al., 2013].

This work introduces participatory search learning, a collection of learning procedures constructed upon evolutionary computation and participatory learning. It introduces a new class of population-based search algorithms using participatory selection, recombination and mutation (variation) operators. The nature of the participatory selection and variation operators are formally studied. In particular, this work focuses on the Participa-

tory Search with Arithmetical Recombination (PSAR), analyses its convergence behavior, and evaluates its performance. PSAR uses compatibility between individuals during selection, recombination and mutation. Jointly, compatibility degrees and objective function shape the evaluation of the fitness of the individuals and govern population evolution. PSAR recombination emerges from an instance of a participatory learning update formula and resembles arithmetical crossover except that it is modulated by a compatibility degree and an arousal index. PSAR mutation depends on compatibility degree and arousal index of participatory learning as well. It arises from a scheme similar to mutation of differential evolution. Performance of the PSAR is evaluated using benchmark optimization problems and compared against the results produced by several population-based algorithms reported in the literature. Computational experiments show that PSAR is, statistically speaking, at least as good as the current best algorithms. The PSAR is further evaluated using actual data of electric maintenance, Auto MPG6, categorical analysis, and abalone age. Computational results show that PSAR outperforms state of the art genetic fuzzy systems [Alcalá et al., 2011], [Antonelli et al., 2013]. PSAR is simpler and produces better solutions faster.

The remainder of the work is organized as follows. Chapter 1 briefly reviews background methodological and theoretical notions, covering evolutionary computation, participatory learning, similarity based approaches, genetic fuzzy systems, and random search techniques. Chapter 2 develops the participatory learning approach and performs a theoretical analysis of the participatory variation operators. It shows that the participatory variation operators induce diversification and intensification. Also, we introduce the random search techniques to study the convergence of participatory search with arithmetical recombination (PSAR). Analysis shows that PSAR converges in probability to global optimum. Chapter 3 evaluates PSAR and compares its performance against several population-based algorithms using classic benchmark optimization problems [Simon, 2008]. The results show that, assuming maximum number of generations as a stop criterion, the PSAR is simpler and finds the optimum solutions for the majority of the problem instances. The PSAR is also compared with the top two performing algorithms of IEEE Congress on Evolutionary Computation (IEEE CEC) 2013 competition. Computational results show that, statistically, PSAR is as good as the best IEEE CEC 2013 algorithms. Chapter 4 addresses applications concerning fuzzy rule-based modeling using real world data. The results show that PSAR outperforms current state of the art genetic fuzzy systems approaches. Finally, Chapter 5 summarizes the contributions and suggests issues for further investigation.

Chapter 2

Publications

Journal

- Liu, Y. L. and Gomide, F. "A Participatory Search Algorithm." Evolutionary Intelligence. Springer, Germany, 2016, Accepted for publication.

Book chapter

- Liu, Y. L. and Gomide, F. "On the Use of Participatory Genetic Fuzzy System Approach to Develop Fuzzy Models." Frontiers of Higher Order Fuzzy Sets. Springer, New York, pp. 67-86, 2015.

International conferences

- Liu, Y. L. and Gomide, F. "Participatory Search Algorithms in Fuzzy Modeling." Proc. World Conference in Soft Computing, May 22-25, 2016, Berkeley, CA, USA.
- Liu, Y. L. and Gomide, F. "Genetic participatory algorithm and system modeling." Proceeding of the fifteenth annual conference companion on Genetic and evolutionary computation conference companion - GECCO, 2013, New York: ACM Press, pp. 1687.
- Liu, Y. L. and Gomide, F. "Fuzzy systems modeling with participatory evolution." IFSA World Congress and NAFIPS Annual Meeting (IFSA/NAFIPS), 2013, Edmonton, AB, Canada, pp. 380-385.
- Liu, Y. L. and Gomide, F. "Evolutionary participatory learning in fuzzy systems modeling." IEEE International Conference on Fuzzy Systems, July 1-10, 2013, Hyderabad, India, pp. 1-8.
- Liu, Y. L. and Gomide, F. "Participatory genetic learning in fuzzy system modeling." IEEE International Workshop on Genetic and Evolutionary Fuzzy Systems, April 15-19, 2013, Singapore, pp. 1-7.

Chapter 3

Background and Literature Review

This chapter gives a brief overview of evolutionary computation and addresses the notions of participatory learning, similarity based computation, genetic fuzzy systems, and random search theory. It reviews evolutionary algorithms, explains the main concepts and notions of participatory learning, the idea of similarity based participatory approaches, and genetic fuzzy systems. A random search technique for real-valued optimization problems and its properties are also reviewed.

3.1 Evolutionary Computation

Evolutionary computation is an area of computer science and engineering that uses ideas from biological evolution, such as reproduction, mutation, recombination, natural selection and survival of the fittest, to solve optimization problems. Evolutionary computation techniques mostly involve metaheuristic optimization procedures. The area includes genetic algorithms, evolution strategies, differential evolution, ant colony optimization, particle swarm optimization , biogeography-based optimization [Michalewicz, 1996], [Eiben and Rudolph, 1999], [Price et al., 2006], [Simon, 2008], [Eberbach, 2005] and many other bioinspired procedures.

Ant colony optimization (ACO) is an algorithm inspired by the pheromone deposition of ants. ACO aims at searching for an optimal path in a graph, based on the behavior of ants seeking a path between their colony and a source of food. The original idea has been diversified to solve a wider class of numerical problems, and as a result, several alternatives have emerged, drawing on various aspects of the behavior of ants [Colorni et al., 1991] and [Dorigo, 1992]. Algorithm 1 summarizes the main steps of the ACO algorithm [Dorigo et al., 2006].

Algorithm 1 Ant colony optimization

```
1: procedure ACO
2:     repeat Schedule Activities
3:         ConstructAntsSolutions()
4:         UpdatePheromones()
5:         DaemonActions()
6:     until end-Schedule Activities
7: end procedure
```

The main procedure of the ACO manages the scheduling of the three components of ACO algorithms via the schedule activities construct. The ConstructAntsSolutions procedure manages a colony of ants that will be used to build a solution in the graph. The UpdatePheromones procedure is the process by which the pheromone trails are modified. Finally, the DaemonActions procedure is used to implement centralized actions which cannot be performed by single ants [Dorigo et al., 2006].

Biogeography is the study of the geographical distribution of biological organisms. Mathematical models of biogeography describe the migration of species between islands, along with their speciation and extinction [Lomolino et al., 2006] and [Whittaker and Fernández-Palacios, 2007]. Biogeography-based optimization (BBO) was first introduced in 2008 [Simon, 2008]. BBO uses biogeography as a metaphor for evolutionary computation. The key idea of BBO is to use probabilistic information sharing between candidate solutions based on their fitness values [Simon et al., 2011]. The main steps of the BBO algorithm are shown in Algorithm 2 [Simon, 2008].

Algorithm 2 Biogeography-based optimization

```
1: procedure BBO
2:     Initialize
3:     repeat
4:         Evaluation()
5:         Migration()
6:         Mutation()
7:     until termination criteria are met
8: end procedure
```

In Algorithm 2 the Evaluation() procedure computes the fitness for each solution and defines the probability of immigration and emigration. The Migration() procedure calculates both, the immigration and emigration probabilities. Good solutions have high emigration probability and low immigration probability. Bad solutions have low emigration probability and high immigration probability. The Mutation() process performs mutation based on the mutation probability.

Differential evolution (DE) is a method that employs the difference between two solutions to probabilistically adapt a third solution. In other words, the DE is an optimization method that iteratively tries to improve a candidate solution using vector differences to perturb a population of vectors [Price et al., 2006]. The main steps of the DE algorithm are as follows, Algorithm 3 [Storn and Price, 1997].

In Algorithm 3 the Mutation procedure perturbs vectors using the scaled difference of two randomly selected population vectors. DE adds the scaled difference to a third randomly selected population vector. The Recombination procedure uses a crossover value $CR \in [0, 1]$ to produce the trial vector. Finally, the trial vector is evaluated to decide whether or not it should be selected.

Evolution strategy (ES) uses mutation, recombination, and selection applied to a population of individuals containing candidate solutions to iteratively evolve better and better solutions. An ES may allow more than two parents to contribute to an offspring [Michalewicz, 1996]. The main steps of the ES algorithm are summarized in Algorithm 4.

Algorithm 4 first defines the number of parents and children. Next, it performs recombination using the parents to form children, and perform mutation on all the children.

Algorithm 3 Differential evolution

```
1: procedure DE
2:     Initialization
3:     Evaluation
4:     repeat
5:         Mutation()
6:         Recombination()
7:         Evaluation()
8:         Selection()
9:     until termination criteria are met
10: end procedure
```

Algorithm 4 Evolution strategy

```
1: procedure ES
2:     Initialization
3:     Evaluation
4:     repeat
5:         Recombination()
6:         Mutation()
7:         SelectSurvivors()
8:     until termination criteria are met
9: end procedure
```

Finally, it selects individuals to assemble the new population.

Genetic algorithm (GA) is a method for moving from one population of individuals to a new population using a kind of nature-like selection together with the genetic operators of crossover and mutation. The main steps of the GA algorithm are as in Algorithm 5.

Algorithm 5 Genetic algorithm

```
1: procedure GA
2:     Initialization
3:     Evaluation
4:     repeat
5:         Selection()
6:         Recombination()
7:         Mutation()
8:     until termination criteria are met
9: end procedure
```

In Algorithm 5, Selection() chooses, in the current population, candidate individuals for the next population. Recombination() chooses genes from parent chromosomes to create new offspring. Mutation() randomly modifies the offspring.

Probability-based incremental learning (PBIL) is a type of genetic algorithm where the genotype of an entire population (probability vector) is evolved rather than individual members [Gupta, 1999]. PBIL combines elements from evolutionary computation (EC) and reinforcement learning (RL). PBIL is a population-based stochastic search where the population is essentially a random sample based on an estimated probability distribution

for each variable. The main steps of the PBIL algorithm are given in Algorithm 6 [Baluja, 1994].

Algorithm 6 Probability-based incremental learning

```
1: procedure PBIL
2:     Initialize probabilities
3:     repeat
4:         GenerateSamples()
5:         FindBest()
6:         UpdateProbability()
7:         MutateProbability()
8:     until termination criteria are met
9: end procedure
```

In Algorithm 6, the GenerateSamples() procedure generates the samples using the current probability matrix and selects the best sample. In UpdateProbability() procedure updates the matrix of probabilities using the best sample to guide the direction of probability update. MutateProbability() performs the mutation rule probabilistically.

Stud genetic algorithm (SGA) is a genetic algorithm that uses the best individual at each generation for crossover [Khatib and Fleming, 1998]. The main steps of the SGA algorithm are summarized in Algorithm 7 [Khatib and Fleming, 1998].

Algorithm 7 Stud genetic algorithm

```
procedure SGA
    Initialize population
    repeat
        Selection()
        Crossover()
        Mutation()
    until termination criteria are met
end procedure
```

In Algorithm 7, the Selection() procedure chooses the fittest individual for mating. The Crossover() procedure uses the best individual in the population to mate with all others and to produce the new offspring. The mutation is the standard bit mutation with low probability.

Particle swarm optimization (PSO) is inspired by social behavior. PSO optimization uses a population of candidate solutions, called particles, and moves these particles in the search-space according to their position and velocity. Each particle movement is influenced by its local best known position and is also guided toward the best known positions in the search-space, which are updated as better positions are found by other particles. The idea is to move the swarm toward the best solutions [Eberhart and Kennedy, 1995] and [Kennedy et al., 2001]. The main steps of the PSO algorithm are as in the Algorithm 8 [Kennedy et al., 1995].

The PSO algorithm 8 starts choosing particles with random position and velocity vectors. The EvaluateFitness() procedure calculates fitness values to choose the particle with the best solution (pbest). The CalculateVelocity() and UpdatePosition() procedures

Algorithm 8 Particle swarm optimization

```
1: procedure PSO
2:     Initialize particles
3:     repeat
4:         EvaluateFitness()
5:         Selection()
6:         CalculateVelocity()
7:         UpdatePosition()
8:     until termination criteria are met
9: end procedure
```

change the velocity and location of each particle toward the best solution, randomly weighting the acceleration at each step.

3.2 Participatory Learning

Search algorithms are at the crossroads of many important areas of increasing relevance for the current technological scenario. These include artificial intelligence [Russell and Norvig, 2003], evolutionary computation [Eiben and Smith, 2015], swarm intelligence [**?**], learning [Sra et al., 2012], and optimization [Simon, 2013]. In computer science, search algorithms use the simplest method of searching through the search space, whereas informed search algorithms use heuristic functions to apply knowledge about the structure of the search space to try to reduce the amount of time spent searching [Meghanathan et al., 2012]. Evolutionary computation use ideas from evolution as heuristics for reducing the search space.

In the early 1990s Yager suggested participatory learning as a scheme in which the process of learning depends on what is already known or believed [Yager, 1990]. A central characteristic of the idea of participatory learning is that an observation has the greatest impact in causing learning or knowledge revision when it is compatible with the current knowledge. Learning occurs in an environment in which the current knowledge participates in the process of learning about itself. Therefore, a fundamental part of this learning scheme is the compatibility between observation and knowledge. The implication of this in search algorithms is that every aspect of the search process is affected and guided by the individuals of a population S^t at step t. The name emphasizes the fact that during search we are in an environment in which the current state of the search process, as mirrored by current population S^t, participates in the process of the search itself. In particular, participatory search uses the compatibility between individuals of the population S^t to direct the steps of the search process via next population S^{t+1}. The notion of compatibility and similarity are recurrent in machine learning, computational intelligence, and evolutionary computation.

We anticipate that the aim of this work is to introduce participatory search algorithms (PSA), population-based search procedures derived from the participatory learning paradigm [Yager, 1990]. There are four main instances of PSA, respectively, participatory search with selective transfer (PSST), participatory search with arithmetical recombination (PSAR), differential participatory search with selective transfer (DPST), and differential participatory search with arithmetical recombination (DPSA). They are distinguished by the nature of the recombination operation, and the order in which the

operations of selection, recombination, and mutation are processed at each step. In PSA the compatibility between individuals affects selection, recombination and mutation at every algorithm step. The emphasis will be in PSAR where, as it will be shown shortly, recombination resembles arithmetical crossover except that it emerges from an instance of a participatory learning formula. The effect of recombination is modulated by compatibility and arousal indexes. PSAR mutation is a form of differential variation as in differential evolution weighted by compatibility and arousal indexes.

More specifically, this work introduces a new class of population-based search algorithms based on participatory learning. In common with other types of evolutionary algorithms, participatory search operates with a population of solutions, rather than with a single solution at a step, and employs procedures to combine these solutions to create new ones. Participatory search is a novel instance of search algorithm because it violates the premise of most evolutionary approaches in that they must necessarily be based on randomization [Fogel, 1998], [Glover et al., 2000] though it adds a randomization step as a diversification mechanism. Participatory search algorithms embody principles that are still not used by other evolutionary approaches, and that prove advantageous to solve a variety of complex optimization problems. Distinct types of participatory search algorithms have been reported [Liu and Gomide, 2016]. They differ in the order in which selection, recombination, and mutation are performed at each algorithm step, and in the type of the recombination operation they use. In particular, PSAR performs convex combinations modulated by the compatibility during recombination, followed by differential variation modulated by compatibility during mutation.

3.3 Similarity Based Approaches

Many similarity-based schemes have been reported in the literature whose aim is to enhance the performance of population-based algorithms target for optimization problems. This section reviews evolutionary approaches based on similarity.

Similarity-based mating was suggested by Ishibuchi and Shibata [Ishibuchi and Shibata, 2003b]. The idea of similarity-based mating is to recombine similar parents. This mating scheme was extended to recombine extreme and similar parents in [Ishibuchi and Shibata, 2003a]. Similarity-based mating mechanisms have been explored and successful results have been achieved. For instance, [Ishibuchi and Shibata, 2003a] select two sets with a specified number of randomly chosen individuals. The farthest individual from the average fitness of the individuals of the first set is selected as the first parent. Next, the most similar to the farthest is selected among the individuals of the second set of individuals as the second parent. The idea is to avoid recombination of dissimilar mates from which good offspring are not likely to be created. Experimental results show that similarity-based mating improves the performance of evolutionary algorithms for multi-objective combinatorial optimization problems.

In [Fonseca et al., 2009] similarity is used as a similarity-based surrogate mechanism to enhance performance of genetic algorithms. Similarity-based surrogate models (SBSMs) belong to the class of lazy learners (memory-based learners). SBSMs simply store their inputs and defer processing until a prediction of the fitness value of a new individual is requested. Then they reply by combining their stored data using a similarity measure, and discard the constructed answer as well as any intermediate results [Fonseca et al., 2009]. The similarity-based surrogate model is based on the k-nearest neighbors method (K-NN). Numerical experiments reported show that the framework is an attractive alternative in

applications that require expensive fitness function evaluations.

In many real-coded genetic algorithms (r-CGA), crossover and mutation operations are performed using similarity between individuals instead of given probability [Chen and Wang, 2010], where similarity is based on the difference-degree between a pair of chromosomes. The difference degree is calculated as the ratio between the number of distinct genes and the total number of genes. The idea has produced efficient real-coded genetic algorithms [Chen and Yin, 2012].

3.4 Genetic Fuzzy Systems

This section briefly overviews genetic fuzzy systems (GFS) and applications. More specifically, we address genetic fuzzy rule-based systems (GFRBS), one of the most important types of GFS. The structure of GFRBS is shown in Figure 3.1.

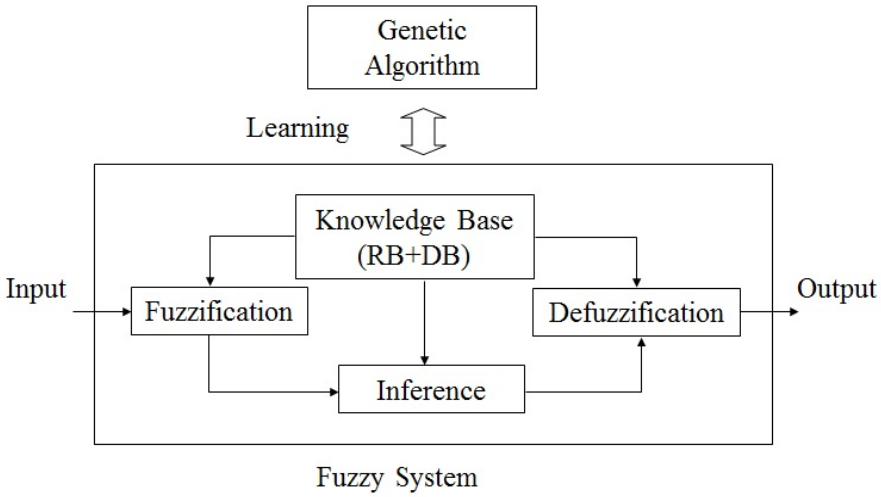


Figure 3.1: Genetic fuzzy rule-based system.

GFRBS is a fuzzy rule-based system enhanced by a learning procedure based on genetic algorithms. A fuzzy rule-based system (FRBS) has a knowledge base (KB) that encodes the knowledge of a target model. The KB contains two main components, a data base and a fuzzy rule base. The data base (DB) stores the linguistic variables used by the fuzzy rules, the membership functions that define the semantics of the linguistic labels, and the parameters of the model. The fuzzy rule base (RB) is a collection of fuzzy if-then rules. Other three components complete fuzzy rule-based models. The first is a fuzzification module to serve as an input interface with the fuzzy reasoning process. The second is an inference engine to perform fuzzy reasoning. The third is a defuzzification output interface module to convert a fuzzy output into a representative pointwise output. An effective approach to construct the KB of an FRBS is to simultaneously develop the DB and the RB within the same process, but in two steps such as in embedded GFRBS learning. Embedded GFRBS is a scheme to learn the DB using simultaneously a simple method to derive a RB for each DB.

The embedded GFRBS, however, does not necessarily provide simple, transparent, and competitive models in terms of the generalization capability. Also, they do not scale

well in terms of processing time and memory, two essential requirements to handle high-dimensional, large-scale, and complex problem solving efficiently [Alcalá et al., 2011]. These issues are addressed in [Alcalá et al., 2011] suggesting a way to reduce the search space in an embedded genetic DB learning framework and fast multiobjective evolutionary algorithm. Lateral displacement of fuzzy partitions using a unique parameter for all membership functions of each linguistic variable is one of the mechanisms the authors adopt to reduce search space dimension. The idea is to prescreen promising partitions to avoid overfitting and maintain coverage and semantic soundness of the fuzzy partitions. In addition, the evolutionary algorithm includes incest prevention, restarting, and rule-cropping in the RB generation process to improve convergence and learning. Despite the use of a mechanism to manage dimensionality, the algorithm does not scale up on the number of data in datasets. The mechanism to deal with scalability is to avoid large percentage of samples, and error estimation using a reduced subset. A post-processing step further refines the algorithm.

Application examples of GFS are many. For instance, [Voget and Kolonko, 1998] presents a multi-objective optimization in which a fuzzy controller regulates the selection procedure and fitness function of genetic algorithms. This approach is used to develop timetables of railway networks aiming at reducing passenger waiting time when switching trains, while at the same time, minimizing the cost of new investments to improve the necessary infrastructure. The result of the genetic optimization is a cost-benefit curve that shows the effect of investments on the accumulated passenger waiting time and trade-offs between both criteria. In [Hwang, 1998] the aim is to optimize trip time and energy consumption of a high-speed railway with fuzzy c-means clustering and genetic algorithm. The method is applied to derive a control strategy for a planned high-speed train line. An economical train run with a trip time margin of less than 7% and an energy saving of 5% is reported. A model to relate the total length of low voltage line installed in a rural town with the number of inhabitants in the town and the mean of the distances from the center of the town to the three furthest clients in it is discussed in [Cordón, 2001]. The authors compare the training and test set error achieved by different modeling techniques for low line estimation.

3.5 Random Search Techniques

Heuristic search algorithms are procedures whose purpose is to find approximate optimal solutions, that is, feasible solutions that are not guaranteed to yield exact optima. Although empirical results indicate that heuristic search algorithm can indeed find good solutions to complex problems, there are few theoretical results concerning their convergence properties [Rudolph, 1994]. In random search theory, an algorithm is said to converge to the global optimum if it generates a sequence of feasible solutions or functions values in which the global optimum is a limit value [Klenke, 2013].

Random search techniques offer an appropriate model for analyzing optimization problems and they have been used in [Karnopp, 1963] and [Solis and Wets, 1981] to address probabilistic convergence behavior of the best solution of a population to the global optimum under elitist selection. That is, the best individual survives with probability one. Convergence of the participatory search algorithm can be approached within random search theory. Here we review the notion of random search and establish a connection between its limiting behavior and global search. The idea is to obtain conditions for which the participatory search algorithm converges to the global optimum. The condi-

tions needed are based on a conceptual algorithm described next. The purpose of the algorithm is to solve the following problem:

Problem P: Given a function f from $\mathbb{R}^n$ to $\mathbb{R}$, and S a subset of $\mathbb{R}^n$, find s in S which minimizes f on S or at least which produce an acceptable approximation of the infimum of f on S.

Thus, the problem is to find the global minimum. Sometimes, the global minimum may not exist or the minimum occurs at point in which f is singularly discontinuous. To avoid these pathological situations, the search for the infimum is replaced by the search for the essential infimum of f on S, and characterization of the optimality region, as suggested in [Solis and Wets, 1981].

Definition 1. *σ is the essential infimum of f on S, defined as follows:*

$$\sigma = inf\{t : \upsilon[s \in S | f(s) < t] > 0\}, \tag{3.1}$$

where $\upsilon(A)$ denotes n-dimensional volume of the set A, υ is the Lebesgue measure.

In measure theory, the Lebsegue measure is the concrete way to measure the subsets of n-dimensional Euclidean space. For $n = 1, 2$, or 3, it coincides with the measure of length, area, or volume. For simplicity, it is called n-dimensional volume or simply volume.

Definition 1 means that the set of points s that produce values $f(s)$ close to the essential infimum σ has nonzero υ-measure, where υ is a nonnegative measure defined on subsets $\mathcal{B}$ of $\mathbb{R}^n$ with $\upsilon(S) > 0$. Therefore, we avoid the case in which the infimum of the function occurs at a point in which it is singularly discontinuous.

For example, let $f(s) = s^2$ when $s \neq 1$ and $f(1) = -10$. Thus $inf(f) = -10$ with $s = 1$. The essential infimum of function is 0.

To avoid the issue of nonexistence of a global minimum, we assume that there exists an optimality region circumscribing the candidates for a minimum [Royden and Fitzpatrick, 1988].

Definition 2. *Optimality region for P is given by*

$$R_{\epsilon,M} = \begin{cases} \{s \in S | f(s) < \sigma + \epsilon\} & if\ \sigma \quad is \quad finite, \\ \{s \in S | f(s) < M\} & if\ \sigma = -\infty. \end{cases}$$

where $\epsilon > 0$ and $M < 0$.

Definition 2 means that whenever a point s is produced such that it is in the optimality region R, then s is considered an optimal solution.

The algorithm in the following serves as a general conceptual random search algorithm (RSA).

Algorithm 9 Conceptual random search algorithm

```
1: procedure RSA
2:     Start with a solution s^0 in S and let t = 0
3:     repeat
4:         Generate ξ^t from the sample space (ℝ^n, ℬ, μ_t)
5:         Set s^{t+1} = D(s^t, ξ^t), choose μ_t, set t = t + 1
6:     until termination criterion is met
7: end procedure
```

Referring to conceptual algorithm, D is a mapping that combines the new sample, ξ^t, with the current solution, s^t. The algorithm produces new solutions according to D, which means that newly produced ones are no worse than the current best solution.

Let M_t be the support of the probability measure μ_t. That is, M_t is the smallest closed subset of $\mathbb{R}^n$ with measure 1. Almost all random search algorithms are adaptive, with μ_t depending on the previous solutions, $s^0, ..., s^{t-1}$ generated by the algorithm. Thus μ_t may be viewed as conditional probability measure.

The following assumptions are required to prove global convergence [Solis and Wets, 1981].

Assumption 1: The map D with domain $S \times \mathbb{R}^n$ and the nonincreasing sequence $\{f(s^t)\}_{t=1}^{\infty}$ are such that

$$f(D(s, \xi)) \leq f(s) \tag{3.2}$$

and

$$\xi \in S \Rightarrow f(D(s, \xi)) \leq min\{f(\xi), f(s)\}. \tag{3.3}$$

Here, convergence means with probability 1 to obtain a monotone sequence $\{f(s^t)\}_{t=1}^{\infty}$ which converges to the infimum of f on S.

Assumption 2: For any subset A of S with $\upsilon(A) > 0$ we have that

$$\prod_{t=0}^{\infty}(1 - \mu_t(A)) = 0. \tag{3.4}$$

This assumption means zero probability of repeatedly missing any positive volume subset of S. In other words, the sampling strategy given by μ_t cannot consistently ignore a part of S with positive volume $\sigma > 0$. In the case of global search algorithms, the Assumption 2 is sufficient to prove convergence to a global minimum. We introduce the global search convergence theorem as follows.

Theorem 1. *Suppose that f is a measurable function, S is a measurable subset of $\mathbb{R}^n$ and that (Assumption 1) and (Assumption 2) hold. Let $\{s^t\}_{t=0}^{\infty}$ be a sequence generated by the conceptual random search algorithm. Then*

$$\lim_{t \to \infty} P\left(s^t \in R_{\epsilon,M}\right) = 1 \tag{3.5}$$

where $P\left(s^t \in R_{\epsilon,M}\right)$ is the probability that at step t, the points s^t generated by the algorithm is in $R_{\epsilon,M}$.

The proof of the theorem is found in [Solis and Wets, 1981].

3.6 Summary

This chapter has reviewed the notions of evolutionary computation, participatory learning, similarity based search, genetic fuzzy systems, and random search theory. They are necessary ingredients to develop the participatory search algorithms and to derive their properties. Next chapter details the participatory search algorithms, analyses the operators they use and the convergence of the participatory search algorithms.

Chapter 4

Participatory Search Algorithms

This chapter introduces a class of the population-based participatory search algorithms. It starts showing the role of the participatory learning idea and concepts play in the participatory search algorithm. The participatory search operators, respectively, selection, recombination, and mutation, are explained and the role of compatibility and arousal justified. Next, we develop the convergence analysis of participatory search algorithm using the random search theory. An explanation on how participatory search algorithm works is given using a simple optimization problem as a vehicle.

4.1 Participatory Search Learning

Learning is the act of acquiring new or modifying an existing knowledge or belief. It may involve different types of information. The ability to learn is possessed by humans, and machines sometimes. Usually learning is a multi-step build up procedure weighted by what is already known. Learning may be viewed as a process that produces changes in the environment and changes to itself. In particular, in the realm of the survival of the fittest mechanism it seems that an additional process occurs. External requirements and the population itself strongly affect reproduction.

A population affects reproduction of its individuals through interaction, compatibility, and imitation among the individuals themselves. It seems natural to investigate the possibility of constructing population-based algorithms that account for the role the population plays in its own reproduction and evolution. Reproductive suitability is the joint effect that results when an individual determines its fitness using its own aims and its compatibility with the remaining individuals. In natural environments, often the effect of the population participation in the crafting fitness is to try to make fitness more like itself [Yager, 2000].

In many situations, learning is a form self-sustaining process in the sense that we learn and revise our beliefs in the context of what we already know or believe. Participatory learning (PL) is a learning paradigm of this type. The name emphasizes that our current knowledge of what we are trying to learn participates in the process of learning about itself [Yager, 1990]. A key property of participatory learning is that input information has the greatest impact in causing learning or knowledge revision when it is compatible with the current knowledge. A fundamental factor of participatory learning is the compatibility degree between input information and current knowledge. Figure 4.1 summarizes the idea. The current knowledge, denoted by $v(t)$, in addition to provide a standard against

which input information $z(t)$ is compared via the lower loop, directly affects the learning process. The upper right loop indicates that current knowledge affects how the system accepts and processes input information. It corresponds to the participatory nature of learning process. High compatibility between the current knowledge and current input information enhances the environment for learning. In PL, this enhancement is expressed by a compatibility degree. Several questions can be raised about the PL structure. One question is that no facility is provided to measure the confidence we have in the current knowledge structure, as we proceed as if we have complete confidence in our current knowledge. If a long sequence of inputs have low compatibility with current knowledge, we may believe that what has been learned so far is wrong, not the observations. This is seen as a form of arousal. Participatory learning introduces an arousal mechanism to monitor the performance of the learning process by looking at the values of the compatibility degree of the current knowledge with inputs. Monitoring information is feedback, via the upper arousal loop of Figure 4.1, in terms of an arousal index that subsequently affects the learning process.

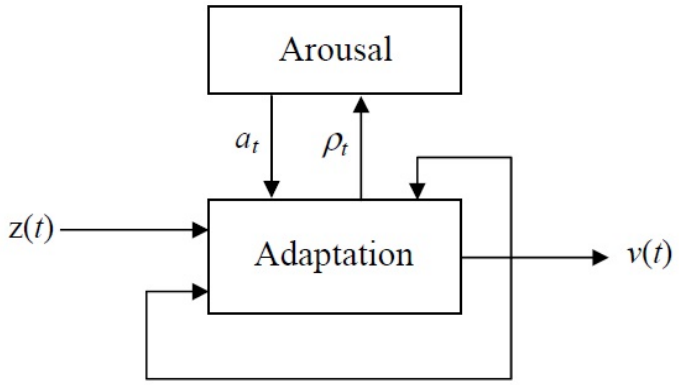


Figure 4.1: Participatory learning.

The instance of participatory learning we adopt in this works uses the compatibility degree between current knowledge and current input to update knowledge employing a smoothing like procedure [Brown, 2004], [Yager, 1990] as follows

$$v(t+1) = v(t) + \alpha\rho_t(z(t) - v(t)) \tag{4.1}$$

where $v(t)$ and $z(t)$ are n-dimensional vectors that denote the current knowledge and current input, respectively. We assume, without loss of generality, that $v(t), z(t) \in [0,1]^n$. The parameter $\alpha \in [0,1]$ is the basic learning rate and $\rho_t \in [0,1]$ is the compatibility degree between $v(t)$ and $z(t)$ at step t. The product of the basic learning rate by the compatibility degree produces the effective learning rate. If an input is far from the current knowledge, then the value of the corresponding compatibility degree is small and the input is filtered once the effective learning rate is lowered by the compatibility degree. This means that if input data are too conflicting with the current knowledge, then they are discounted [Yager, 1990]. Low values of effective learning rate avoid swings due to spurious values of input information which are far from current knowledge. Expression (4.1) corresponds to the lower feedback loop of Figure 4.1. As it will be shown in the

next subsection, (4.1) plays a key role in this work because it produces the recombination operator of PSA.

The mechanism to monitor compatibility degrees during learning is the arousal index. We can introduce arousal in the basic PL knowledge update formula (4.1) as follows

$$v(t+1) = v(t) + \alpha\rho_t^{1-a_t}(z(t) - v(t)) \tag{4.2}$$

where $a_t \in [0,1]$ is the arousal index at t.

One way to compute the compatibility degree ρ at step t is

$$\rho_t = 1 - \frac{1}{n}\sum_{k=1}^{n} |z_k(t) - v_k(t)|. \tag{4.3}$$

In (4.3) ρ_t is the complement of the average absolute difference between input information $z(t)$ and current belief $v(t)$. In a more general sense, ρ_t may be seen to be a measure of similarity between $z(t)$ and $v(t)$. If $\rho_t = 0$, then $v(t+1) = v(t)$ and the current input $z(t)$ is completely incompatible with the current knowledge $v(t)$. This condition means that the system is not open to any learning from the current information. On the other hand, if $\rho_t = 1$, then $v(t+1) = z(t)$. In this case input information is in complete agreement with the current knowledge and the system is fully open to learn.

A metric is a function that computes the distance between pairs of elements of a set. It is shown next that ρ is such that, for any $x, y, z \in [0,1]^n$

1. $\rho(x,y) \geq 0$
2. $\rho(x,y) = 0 \Leftrightarrow x = y$
3. $\rho(x,y) = \rho(y,x)$
4. $\rho(x,z) \leq \rho(x,y) + \rho(y,z)$

that is, ρ is a distance measure. To see this, rewrite ρ as follows

$$\begin{aligned} \rho &= 1 - \frac{1}{n}\sum_{k=1}^{n} |z_k - v_k| && (4.4)\\ &= \frac{1}{n}\sum_{k=1}^{n}(1 - |z_k - v_k|). && (4.5) \end{aligned}$$

Consider the term $1-|z_k-v_k| = S_k$, which can be understood as a measure of similarity between z and v.

For property 1,

$$\rho(z,v) = \frac{1}{n}\sum_{k=1}^{n} S_k \geq 0 \quad \text{because} \quad S_k \in [0,1].$$

For property 2,

$$\rho(z,v) = \frac{1}{n}\sum_{k=1}^{n} S_k = 0 \Rightarrow \sum_{k=1}^{n} S_k = 0, \quad \text{thus,} \quad z = v.$$

On the other hand,

$$z = v \Rightarrow \sum_{k=1}^{n} S_k = 0, \quad \text{thus,} \quad \frac{1}{n}\sum_{k=1}^{n} S_k = 0 = \rho(z, v).$$

For property 3,

$$\begin{aligned} \rho(z, v) &= 1 - \frac{1}{n}\sum_{k=1}^{n} |z_k - v_k| && (4.6)\\ &= 1 - \frac{1}{n}\sum_{k=1}^{n} |v_k - z_k| && (4.7)\\ &= \rho(v, z). && (4.8) \end{aligned}$$

Finally, for property 4, let $z, v, w \in S$,

$$\begin{aligned} \rho(z, w) &= 1 - \frac{1}{n}\sum_{k=1}^{n} |z_k - w_k| && (4.9)\\ &= 1 - \frac{1}{n}\sum_{k=1}^{n} |z_k + v_k - v_k - w_k| && (4.10)\\ &= 1 - \frac{1}{n}\sum_{k=1}^{n} |(z_k - v_k) + (v_k - w_k)| && (4.11)\\ &\leq 1 - \frac{1}{n}\sum_{k=1}^{n} |z_k - v_k| - \frac{1}{n}\sum_{k=1}^{n} |v_k - w_k| && (4.12)\\ &= \rho(z, v) + \rho(v, w) - 1 && (4.13)\\ &\leq \rho(z, v) + \rho(v, w). && (4.14) \end{aligned}$$

A way to model arousal is to see it as the complement of the confidence in the current knowledge. A simple procedure is to update the arousal index a at step t is

$$a_{t+1} = (1 - \beta)a_t + \beta(1 - \rho_{t+1}) \tag{4.15}$$

where $\beta \in [0, 1]$ controls the rate of change of arousal. The higher a_t, the less confident is the learning system in current knowledge. If $\rho_{t+1} = 1$, then we have a highly compatible input and the arousal index decreases. On the other hand, if $\rho_{t+1} = 0$, then input information compatibility is low and the arousal index increases.

The notion of compatibility degree contributes with PSA to form pools of individuals for selection, recombination, and mutation. The pools are assembled by two populations S^t and $S^{t'}$. The individuals of $S^{t'}$ are those of S^t which are the most compatible, one to one. Selection uses compatibility to select individuals of the pool which are closer with current best individual. Recombination is done pairwise between individuals of the mating pool, modulated by their compatibility and arousal indexes. Mutation proceeds by adding to the current best individual a variation proportional to the difference between selected and recombined individuals modulated by the corresponding compatibility indexes.

In analogy with the participatory learning paradigm, the current population and the current best individual play the role of *current knowledge*, whereas the new population

plays the role of the *new knowledge*. The *learning object* is the individual whose value, as measured by an objective/fitness function, is the highest. The effect of compatibility in PSA is to encourage selection and recombination of similar mates from which good offspring are likely to be produced.

A class of participatory search algorithms that incorporates participatory learning is shown in Figure 4.2. There are four instances, respectively, participatory search with selective transfer (PSST), participatory search with arithmetical recombination (PSAR), differential participatory search with selective transfer (DPST), and differential participatory search with arithmetical recombination (DPSA). They are distinguished by the nature of the recombination, and the order in which the operations of selection, recombination, and mutation are processed in each generation. They also differ from similar evolutionary approaches developed in [Liu and Gomide, 2013c] [Liu and Gomide, 2013a] and [Liu and Gomide, 2013b] in the way the mating pool is constructed to produce the new population.

PSST is similar to the algorithm introduced in [Liu and Gomide, 2013c] in that both use participatory selective transfer and mutation. PSAR uses participatory arithmetical recombination and mutation, processed in a different order than PSST. DPST is similar to the algorithm of [Liu and Gomide, 2013a] because it also uses selective transfer and participatory mutation. Likewise, DPSA is similar to the algorithm of [Liu and Gomide, 2013b] and uses participatory arithmetical recombination and mutation. DPSA proceeds similarly as DPST except that it uses arithmetical recombination instead of selective transfer. PSST, PSAR, DPST and DPSA differ from all previous approaches because selection is done for each of the N individuals of the current population. Participatory recombination and mutation are performed likewise. PSST, PSAR, DPST and DPSA are all elitist: the best individual is always kept in the current population.

The participatory search algorithms (PSA) will be detailed in Section 2.4.

Earlier computational evaluations of PSAR performs best amongst PSST, DPSA and DPST have been performed and the results, summarized in [Liu and Gomide, 2016], show that PSAR performs best among the remaining ones. Therefore, performance evaluation, comparisons with alternative population-based algorithms, and convergence analysis will be focused on PSAR only. All the participatory search operators and their properties are, however, detailed next.

4.2 Participatory Search Operators

This section addresses the participatory search operators introduced in this work and studies their properties. The study reveals their exploitation, exploration behavior, items of utmost importance in population-based search algorithms.

4.2.1 Selection

Let S be a set of N strings of fixed length n, and $s, s^{'} \in S$ be two individuals, $s^{'}$ distinct of s, such that

$$s' = argmax_{r \in S}(\rho(s, r)) \tag{4.16}$$

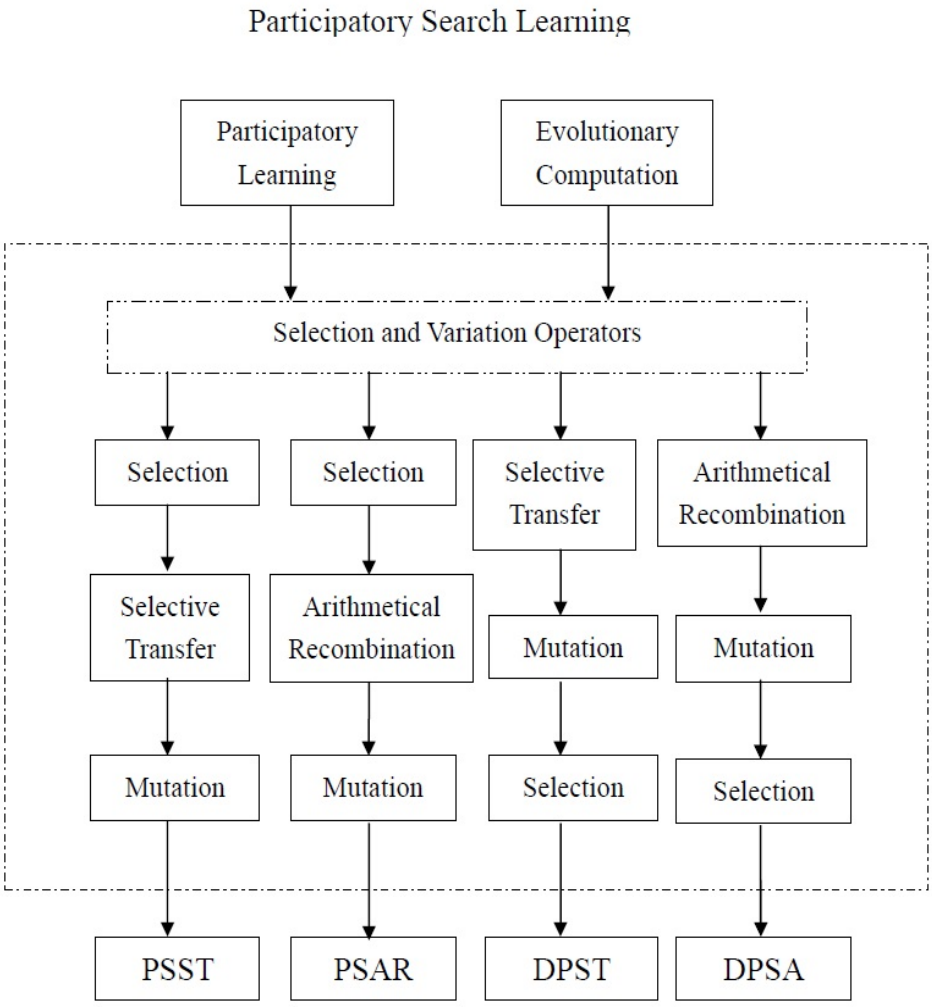


Figure 4.2: Participatory search algorithms (PSA).

where

$$\rho(s,r) = 1 - \frac{1}{n}\sum_{k=1}^{n}|s_k - r_k|, \tag{4.17}$$

and $s = (s_1, s_2, .., s_N)$ and $r = (r_1, r_2, .., r_N)$. The individual $s^{'}$ is the one whose compatibility index with s is the largest. This procedure is repeated for each individual s of S to assemble the pool $S^{'}$ with N individuals. Notice that construction of the pool is biased by the compatibility degrees between the individuals of S. Figure 4.3 illustrates how the population S and S' are assembled.

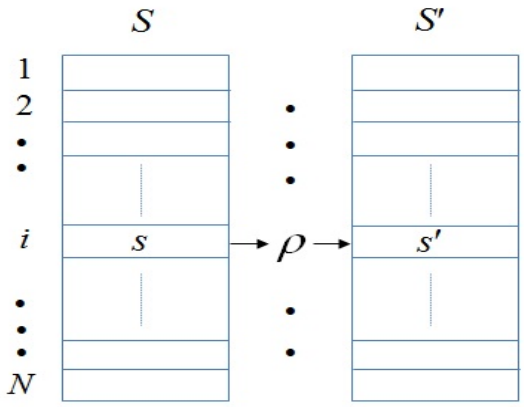


Figure 4.3: A population and its pool.

In PSA, selection is done by computing the compatibility degrees between $s \in S$ and the corresponding $s^{'} \in S^{'}$ with the current best individual $best = s^*$, and picking the one that is the most compatible to assemble a population L of selected individuals, that is, the ones that are the closest to the current best individual. Formally,

$$s^* = argmin_{s \in S} f(s), \tag{4.18}$$

where f is the objective functin.

More specifically, selection computes the compatibility degrees $\rho^s(s, s^*)$ and $\rho^{s^{'}}(s^{'}, s^*)$ using

$$\rho^s = 1 - \frac{1}{n}\sum_{k=1}^{n}|s_k - s_k^*| \tag{4.19}$$

and

$$\rho^{s^{'}} = 1 - \frac{1}{n}\sum_{k=1}^{n}|s_k^{'} - s_k^*|, \tag{4.20}$$

and the individual whose compatibility degree is the largest, denoted by $p_{selected}$, is selected. That is, PSA selection proceeds as follows

$$\text{if} \quad \rho^s \geq \rho^{s^{'}} \quad \text{then} \quad p_{selected} = s \quad \text{else} \quad p_{selected} = s^{'}. \tag{4.21}$$

Figure 4.4 illustrates the process of selection.

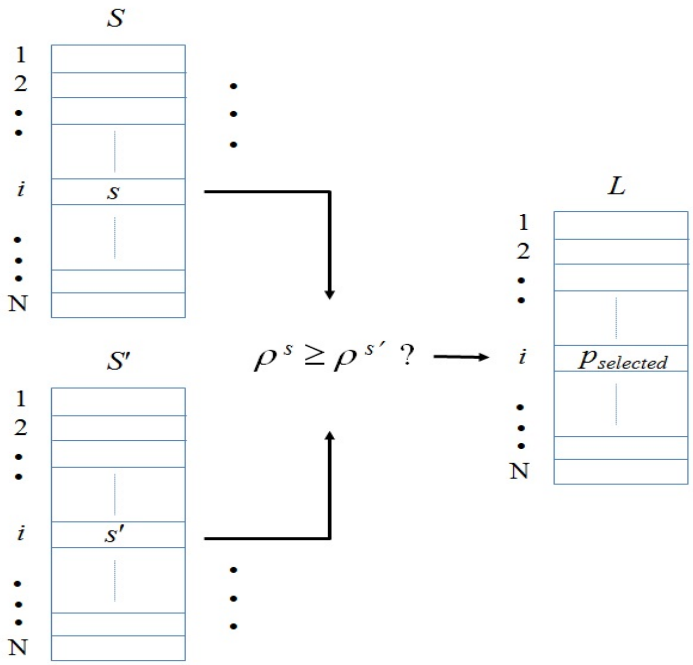


Figure 4.4: Selection.

Selection depends on the objective function $f(s)$, which identifies current best s^*, and on $\rho^s(s, s^*)$ and $\rho^{s'}(s', s^*)$ which measure the compatibility between s^* and the corresponding pair of individuals s and s' of the current pool. Jointly, f, ρ^s and $\rho^{s'}$ decide if an individual is selected or not. Figures 4.5 and 4.6 illustrate the effect of selection when PSA is run using Griewank and Rosenbrock functions [Michalewicz, 1996], respectively. The values of $\rho(p_{selected}, best)$ and $|f(p_{selected}) - f(best)|$ are plotted for several generations. Figures 4.5 and 4.6 show the values of the compatibility $\rho(p_{selected}, best)$ between the selected and best individuals are highly correlated with the respective objective function values: the higher the compatibility, the closer the values of the objective function.

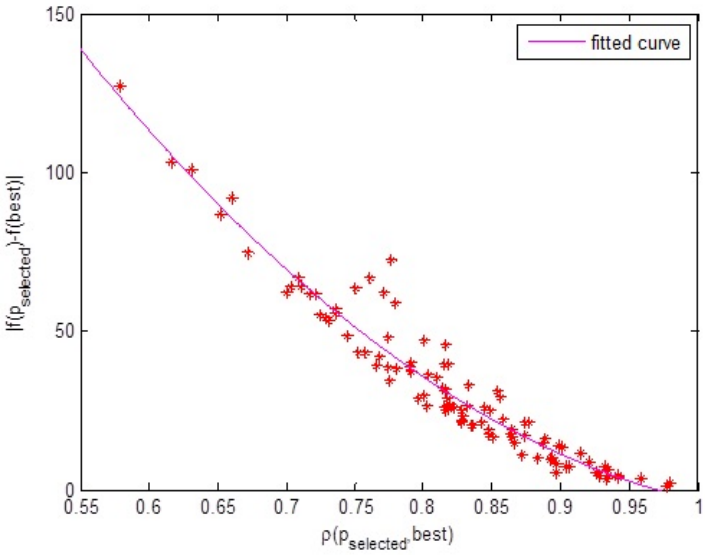


Figure 4.5: f: Griewank function.

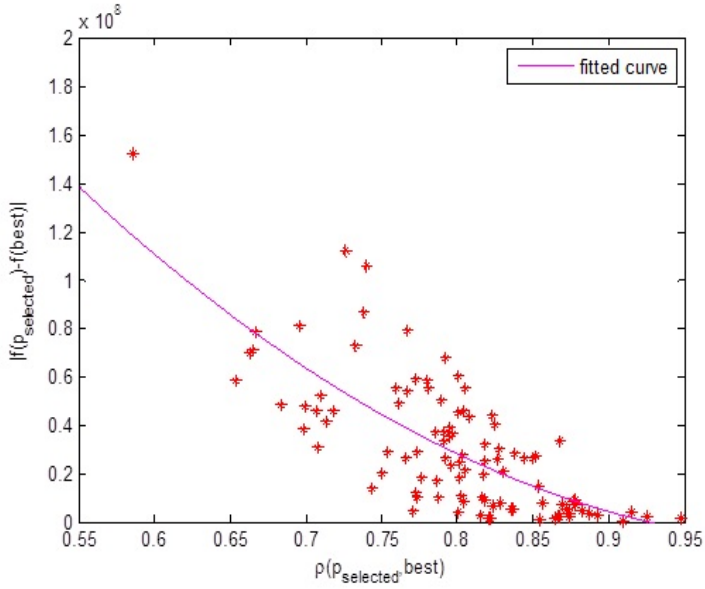


Figure 4.6: f: Rosenbrock function.

4.2.2 Recombination

Recombination is performed by means of two mechanisms: selective transfer and arithmetical recombination. We explain the processes of selective transfer and arithmetical recombination next.

4.2.2.1 Selective Transfer

During the last few years, we have witnessed a growing interest to use economic principles and models of learning in genetic algorithms. For instance, evolutionary pro-

cesses have been used to model the adaptive behavior of a population of economic agents [Birchenhall and shin Lin, 2000]. Here agents develop models of fitness to their environment in conjunction with the corresponding economic activities. Economists believe that behavior acquired through individual experience can be transmitted to future generations, and that learning changes the way to search the space in which evolution operates. This is an argument in favor of the interaction between the processes of evolution and learning. Since technical knowledge is distributed across the economic population, technological change can be viewed as a process of distributed learning. Here, the term learning is used in a broad sense, that is, there is no distinction between learning as propagation of knowledge through the population and the process of innovation, creation, and discovery. The distributed learning perspective helps to understand technological change and focus on the population suggests that an evolutionary perspective may be appropriate.

Birchenhall et al. [Birchenhall and shin Lin, 2000] claim that our knowledge and technology are modular, i.e., they can be decomposed into several components or modules. From the evolutionary computation point of view, they suggest that the crossover operator of genetic algorithms could be seen as a representative of modular imitation. To bring these ideas together, they advocate an algorithm that replaces selection and crossover operators by an operator based on selective transfer. Essentially, selective transfer is a filtered replacement of substrings from one string to another, without excluding the possibility that the entire sequence is copied [Birchenhall et al., 1997]. Clearly, the selective transfer is similar to Holland crossover, but it is one-way transfer of strings, not on exchange of strings. The behavior selective transfer is likely to be very different from the combination of selection and crossover. PSST and DPST translate the selective transfer idea into a recombination procedure as follows.

Assume that an individual $p_{selected}$ is selected using the objective function and compatibility as described in the step of the selection. Two positions $h \leq k$ in the $p_{selected}$ string are chosen randomly, and a fair coin is tossed. If the coin turns head, then the substrings from $p_{selected}(h)$ to $p_{selected}(k)$ of $p_{selected}$ is replaced by the corresponding substrings from $s^*(h)$ to $s^*(k)$ of s^*. If the coin turns up tail, then the substrings from $p_{selected}(1)$ to $p_{selected}(h-1)$ and from $p_{selected}(k+1)$ to $p_{selected}(n)$ are replaced by the corresponding substrings of s^*. Figure 4.7 illustrates the idea of selective transfer.

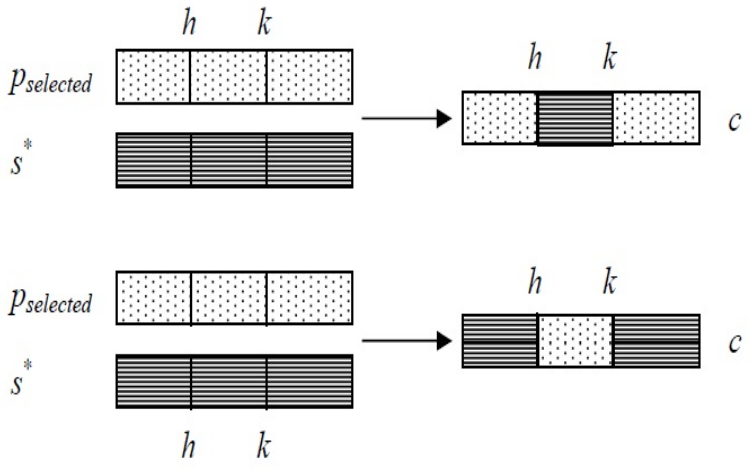


Figure 4.7: Selective transfer.

Despite similarity with crossover of the standard genetic algorithms, there are some differences. The most important one is that selective transfer uses one-way relocation of substrings, from the best individual to the individual selected, and hence it is not a crossover. This is important because selective transfer is much more schemata destructive than the standard crossover [Birchenhall et al., 1997].

4.2.2.2 Arithmetical Recombination

Recombination derives from the participatory learning update formula (4.2). To see this, notice that (4.2) can be rewritten as

$$\begin{aligned} v(t+1) &= v(t) + \alpha\rho_t^{(1-a_t)}(z(t) - v(t)) \\ &= (1 - \alpha\rho_t^{(1-a_t)})v(t) + \alpha\rho_t^{(1-a_t)}z(t). \end{aligned} \tag{4.22}$$

Let $\gamma = \alpha\rho_t^{(1-a_t)}$. Thus (4.22) becomes

$$v(t+1) = (1-\gamma)v(t) + \gamma z(t). \tag{4.23}$$

Expression (4.23) is of the following type

$$s_v(t+1) = (1-\delta)s_v(t) + \delta s_z(t) \tag{4.24}$$

where $\delta \in [0,1]$. Clearly (4.24) is a convex combination of $s_v(t)$ and $s_z(t)$ whose result is the offspring $s_v(t+1)$. Noticeably (4.23) is similar to (4.24) and hence (4.23) is an arithmetical-like recombination. While parameter δ of (4.24) is either a constant or variable, depending on the age of population, the value γ of (4.23) is variable and modulated by compatibility and arousal.

Participatory recombination proceeds as in (4.23) to produce offspring p_r from individuals s and $s^{'}$ of pools S and $S^{'}$, respectively, as follows

$$p_r = (1 - \alpha\rho_r^{(1-a)})s + \alpha\rho_r^{(1-a)}s^{'}. \tag{4.25}$$

Figure 4.8 illustrates the process of participatory recombination. Addition in Figure 4.8 is weighted as in (4.25)

4.2.3 Mutation

There are many ways to do mutation in search algorithms. For example, assume a population of N individuals represented by n-dimensional vectors denoted by $s_{r_i,t}$ at generation t. Differential evolution produces new individuals by adding the weighted differences between distinct vectors to a third vector [Storn and Price, 1997]. For each vector $s_{r_i,t}$, $i = 1, 2, ..., N$, a mutated vector is produced using

$$s_{i,t+1} = s_{r_1,t} + \phi \cdot (s_{r_2,t} - s_{r_3,t}) \tag{4.26}$$

where $r_1, r_2, r_3 \in \{1, 2, ..., N\}$ are random indexes, and $\phi > 0$ is a parameter which controls the amount of the differential variation $(s_{r_2,t} - s_{r_3,t})$.

Mutation in participatory search is similar to differential evolution mutation. It produces a mutated individual p_m as follows

$$p_m = best + \rho_m^{1-a}(p_{selected} - p_r). \tag{4.27}$$

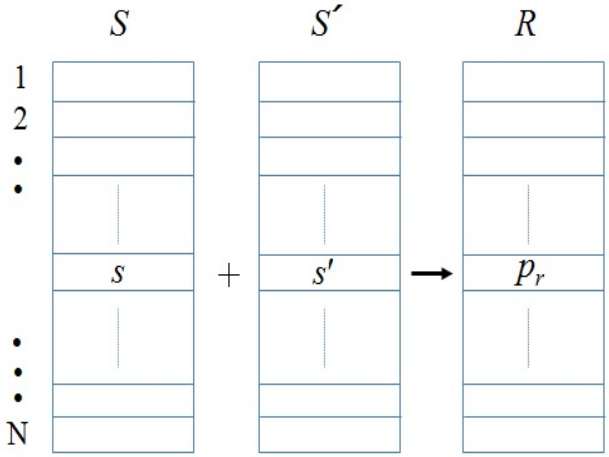


Figure 4.8: Recombination.

Figure 4.9 illustrates the process of mutation.

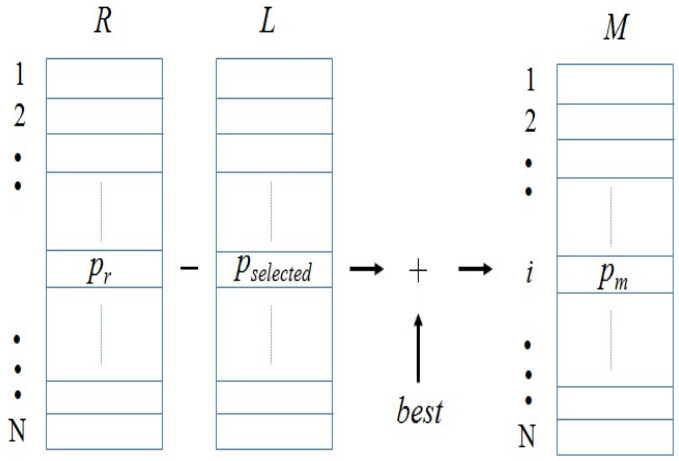


Figure 4.9: Mutation.

In participatory mutation, the amount of the variation of the best individual $best = s^*$ is controlled by compatibility between the selected and recombined individuals and arousal.

4.3 Analysis of the Participatory Operators

This section studies the nature of the participatory search operators. Generally speaking, recombination aims at mixing individuals to combine their properties. Recombination operators should fulfill some design requirements to produce offspring with desirable properties. One condition requires recombination to group together solutions of related fitness [Radcliffe and Surry, 1994]. The idea is to use a set of existing solutions to create a

new solution by recombining the relevant properties of those solutions. Another requirement is to account for the metric used. For example, given two parent solutions s^{p1} and s^{p2} and an offspring s^0, recombination operators should be such that [Rothlauf, 2011]

$$d(s^{p1}, s^{p2}) \geq \max(d(s^{p1}, s^0), d(s^{p2}, s^0)). \tag{4.28}$$

Inequality (4.28) expresses the requirement on the distance of offspring and its parents be equal to or smaller than the distance between the parents. If the distance $d(s^{p1}, s^{p2})$ between s^{p1} and s^{p2} is viewed as a measure of similarity, than (4.28) guarantees that offspring are similar to parents. In a extreme case, the same parents, that is $s^{p1} = s^{p2}$, produce offspring equal to themselves, $s^0 = s^{p1} = s^{p2}$. An example of recombination operator that fulfills (4.28) is the arithmetical crossover [Michalewicz, 1996]. Recall that, given two parents s^{p1} and s^{p2}, the arithmetical crossover (4.24) produces offspring s^0 as follows

$$s^0 = \gamma s^{p1} + (1 - \gamma)s^{p2}, \tag{4.29}$$

where $\gamma \in [0, 1]$. If $\gamma = 0.5$, then offspring is the mean of the parents solutions. From the city-block metric point of view, arithmetical crossover with $\gamma = 0.5$ produces distance between offspring and parents smaller than the distance between the parents [Rothlauf, 2011] and the the similarity between offspring and parents is higher than between parents.

Recombination of PSA resembles the arithmetical crossover, and it should be reasonable to expect (4.28) to hold for participatory recombination as well. Indeed, this is the case as it is shown next.

Theorem 2. *The participatory recombination operator of PSA produces offspring p_r from parents s and s' such that*

$$d(s, s') \geq \max(d(s, p_r), d(s', p_r)).$$

Proof. Recombination of PSA uses parents s and s' to produce offspring p_r as follows

$$p_r = (1 - \gamma)s + \gamma s'$$

where $\gamma = \alpha\rho_r^{1-a}$. Individuals s, s' and p_r play the role of s^{p1}, s^{p2} and s^0 in (4.28). We must show that

$$d(s, s') \geq \max(d(s, p_r), d(s', p_r)). \tag{4.30}$$

We have that

$$\begin{aligned} d(s, p_r) &= d(s, (1 - \gamma)s + \gamma s') \\ &\leq d(s, (1 - \gamma)s) + d(s, \gamma s') \\ &= (1 - \gamma)d(s, s) + \gamma d(s, s') \\ &= \gamma d(s, s') \leq d(s, s') \end{aligned} \tag{4.31}$$

We also have that

$$\begin{aligned} d(s', p_r) &= d(s', (1 - \gamma)s + \gamma s') \\ &\leq d(s', (1 - \gamma)s) + d(s', \gamma s') \\ &= (1 - \gamma)d(s', s) + \gamma d(s', s') \\ &= (1 - \gamma)d(s', s) \leq d(s, s') \end{aligned} \tag{4.32}$$

Because $0 \leq \gamma \leq 1$, (4.31) and (4.32) yield

$$d(s, s') \geq \max(d(s, p_r), d(s', p_r)).$$

□

In what follows, an analysis of the selective transfer behavior is developed.

Theorem 3. *The selective transfer operator of PSA produces offspring c from parents* s^* *and* $p_{selected}$ *such that*

$$d(s^*, p_{selected}) \geq \max(d(s^*, c), d(p_{selected}, c))$$

Proof. Selective transfer proceeds as follows.

(a). choose h, k, $h \leq k$, and $r \in [0, 1]$ randomly.

(b). if $r \leq 1/2$ then $c = [s^*(1:h), p_{selected}(h+1:k), s^*(k+1:n)]$;
else $c = [p_{selected}(1:h), s^*(h+1:k), p_{selected}(k+1:n)]$.

Notice that s^*, $p_{seleted}$ and c play the role of s^{p1}, s^{p2} and s^0 in (4.28). Notation $c = [s^*(1:h), p_{selected}(h+1:k), s^*(k+1:n)]$ means that the string c is assembled merging the substrings $s^*(1:h)$, $p_{selected}(h+1:k)$, and $s^*(k+1:n)$, respectively. Substring $s^*(1:h)$ is a copy of the first h components of string s^*, $p_{selected}(h+1:k)$ is a copy of $p_{seleccted}$ from component $h+1$ up to k. Similarly $s^*(k+1:n)$ is a copy of components from $h+1$ to n of string s^*.

To analyze the behavior of selective transfer, we must verify if

$$d(s^*, p_{selected}) \geq \max(d(s^*, c), d(p_{selected}, c)) \tag{4.33}$$

Because $p_{selected}$ is the individual whose compatibility degree of either s or $s^{'}$ with s^* is the largest, we have two options to be verified, respectively

1. If $p_{selected} = s$, then
$$d(s^*, s) \geq \max(d(s^*, c), d(s, c)) \tag{4.34}$$
2. If $p_{selected} = s'$, then
$$d(s^*, s^{'}) \geq \max(d(s^*, c), d(s^{'}, c)) \tag{4.35}$$

First, to verify (4.34) we need to check if $d(s^*, s) \geq d(s^*, c)$ and $d(s^*, s) \geq d(s, c)$ hold simultaneously. From Figure 4.10, if $r \geq 1/2$, then $d(s^*(1:h), c(1:h)) = 0$, $d(s^*(k+1:n), c(k+1:n)) = 0$, and hence $d(s^*, s) \geq d(s^*, c)$. Similarly, since $d(s(h:k), c(h:k)) = 0$, $d(s^*, s) \geq d(s, c)$. If $r < 1/2$, from Figure 4.10 we see that $d(s^*(h:k), c(h:k)) = 0$, and consequently $d(s^*, s) \geq d(s^*, c)$. Also, because $d(s(1:h), c(1:h)) = 0$ and $d(s(k+1:n), c(k+1:n)) = 0$, $d(s^*, s) \geq d(s, c)$. Thus, (4.34) holds. It can be shown similarly that (4.35) also holds. Therefore, both (4.34) and (4.35) hold. Intuitively speaking, this result suggests that, because the distance of offspring and parents are equal to or smaller than the distance between the parents, selective transfer essentially is an exploitation mechanism.

□

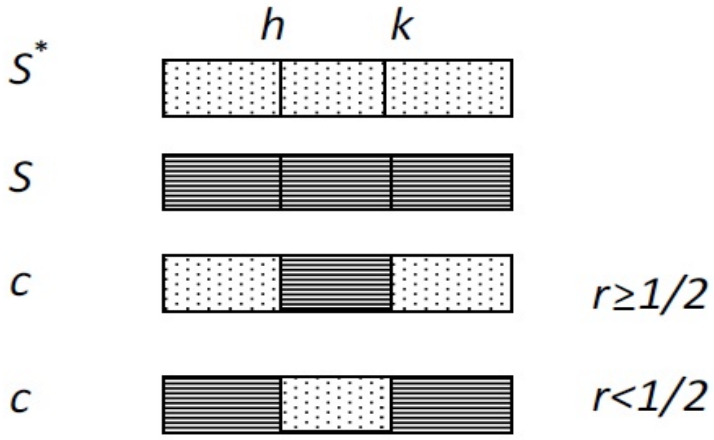


Figure 4.10: Selective transfer

The role of mutation is to induce variability in a population to prevent premature convergence [Blum and Roli, 2003]. In this sense, participatory mutation should encourage exploration of the search space. In participatory mutation, however, an individual may move to closer areas in the search space depending of the amount of variation ($p_{selected} - p_r$). To show that mutation induces exploration is analogous to show that it fulfills the inequality (4.28) reversed. The following theorem shows participatory mutation has this property.

Theorem 4. *The participatory mutation operator of PSA produces p_m from $p_{selected}$ and p_r such that*

$$d(p_{selected}, p_r) \leq \max(d(p_{selected}, p_m), d(p_r, p_m))$$

Proof. PSA mutation uses $p_{selected}$ and p_r to produce p_m from

$$p_m = s^* + \rho_m^{1-a}(p_{selected} - p_r).$$

We have to show that

$$d(p_{selected}, p_r) \leq \max(d(p_{selected}, p_m), d(p_r, p_m)) \tag{4.36}$$

From (4.21) $p_{selected}$ is either s or s'. Assume that $p_{selected} = s$. We must check if

$$d(s, p_r) \leq \max(d(s, p_m), d(p_r, p_m)) \tag{4.37}$$

Indeed, this is the case because

$$p_r = (1 - \gamma)s + \gamma s'$$

where $\gamma = \alpha\rho_r^{1-a}$, we have

$$\begin{aligned} d(s, p_r) &= d(s, (1-\gamma)s + \gamma s') \\ &\leq (1-\gamma)d(s, s) + \gamma d(s, s') \\ &= \gamma d(s, s') \\ &\leq d(s, s') \end{aligned} \tag{4.38}$$

Similarly,

$$\begin{aligned} d(s,p_m) &= d(s, s^* + \delta(s - (1-\gamma)s - \gamma s')) \\ &= d(s, s^* + \delta\gamma(s - s')) \\ &\leq d(s,s^*) + \delta\gamma d(s, s' - s) \\ &\leq d(s,s^*) + \delta\gamma d(s,s') \\ &\leq d(s,s^*) + d(s,s') \end{aligned} \tag{4.39}$$

where $\delta = \rho_m^{1-a}$. Computing $d(p_r, p_m)$ we obtain

$$\begin{aligned} d(p_r,p_m) &= d(p_r, s^* + \delta(s - p_r)) \\ &= d(p_r, s^* + \delta s - \delta p_r) \\ &\leq d(p_r,s^*) + \delta d(p_r,s) - \delta d(p_r,p_r) \\ &= d(p_r,s^*) + \delta d(p_r,s) \\ &\leq d(p_r,s^*) + d(s,s') \end{aligned} \tag{4.40}$$

From (4.38) and (4.39)

$$d(s,p_m) - d(s,p_r) \leq d(s,s^*) > 0 \tag{4.41}$$

and hence $d(s,p_m) > d(s,p_r)$.
From (4.38) and (4.40) we have

$$d(p_r,p_m) - d(s,p_r) \leq d(p_r,s^*) > 0 \tag{4.42}$$

thus $d(p_r,p_m) > d(s,p_r)$. Therefore,

$$d(s,p_r) \leq \max(d(s,p_m), d(p_r,p_m)). \tag{4.43}$$

Assume that $p_{selected} = s'$. We must show that the following inequality holds

$$d(s',p_r) \leq \max(d(s',p_m), d(p_r,p_m)) \tag{4.44}$$

Indeed, calculation of $d(s',p_r)$ gives

$$\begin{aligned} d(s',p_r) &= d(s', (1-\gamma)s + \gamma s') \\ &\leq (1-\gamma)d(s',s) + \gamma d(s',s') \\ &= (1-\gamma)d(s',s) \\ &\leq d(s',s) \end{aligned} \tag{4.45}$$

and calculation of $d(s',p_m)$ gives

$$\begin{aligned} d(s',p_m) &= d(s', s^* + \delta(s' - (1-\gamma)s - \gamma s')) \\ &= d(s', s^* + \delta(1-\gamma)(s' - s)) \\ &\leq d(s',s^*) + \delta(1-\gamma)d(s', s' - s) \\ &= d(s',s^*) + \delta(\gamma - 1)d(s',s) \\ &\leq d(s',s^*) + d(s',s) \end{aligned} \tag{4.46}$$

Computing $d(p_r, p_m)$ we obtain

$$\begin{aligned} d(p_r, p_m) &= d(p_r, s^* + \delta(s' - p_r)) \\ &= d(p_r, s^* + \delta s' - \delta p_r)) \\ &\leq d(p_r, s^*) + \delta d(p_r, s') - \delta d(p_r, p_r) \\ &= d(p_r, s^*) + \delta d(p_r, s') \\ &\leq d(p_r, s^*) + d(s', s) \end{aligned} \tag{4.47}$$

From (4.45) and (4.46) we have

$$d(s', p_m) - d(s', p_r) \leq d(s', s) > 0 \tag{4.48}$$

and hence $d(s', p_m) > d(s', p_r)$.
From (4.45) and (4.47) we get

$$d(p_r, p_m) - d(s', p_r) \leq d(p_r, s^*) > 0 \tag{4.49}$$

thus $d(p_r, p_m) > d(s', p_r)$. Therefore,

$$d(s', p_r) \leq \max(d(s', p_m), d(p_r, p_m))$$

which means that

$$d(p_{selected}, p_r) \leq \max(d(p_{selected}, p_m), d(p_r, p_m)).$$

□

Theorems 3 and 4 identify the requirements the search operators need to produce intensification and diversification. Diversification means to generate diverse solutions so as to explore the search space on the global scale, while intensification means to focus on the search in a local region by exploiting the information that a current good solution is found in this region [Blum and Roli, 2003]. Thus, while participatory mutation induces a diversification mechanism by an exploration of the search space, the participatory recombination and selective transfer have an intensification nature. This means that the participatory search operators may quickly identify regions in the search space with high quality solutions, and try to reduce the amount of time spent in searching.

4.4 Convergence Analysis of Participatory Search

Algorithms 10-13 detail the participatory search procedures PSAR, PSST, DPSA, DPST. We assume S^t as a set of N with strings of length n at step t.

In particular, the PSAR works as follows. It starts a population S^t at $t = 0$ with N randomly chosen individuals and, for each individual of S^t, the most compatible individual amongst the remaining ones is chosen to assemble the population $S^{t'}$ with N individuals. S^t and $S^{t'}$ form the mating pool. Next step computes the best individual s^* in the current population S^t, denoted by *best*. For instance, for minimization problems *best* is such that

$$best = argmin_{s \in S^t} f(s). \tag{4.50}$$

Selection proceeds by selecting, amongst each individual of S^t and the corresponding mate in $S^{t'}$, the one which is closest to *best*. Recombination is done pairwise between the

individuals of the mating pool, weighted by the values of compatibility and arousal. Mutation uses the selected and recombined individuals to produce variations whose amount is weighted by compatibility and arousal. If a offspring is better than current best individual, then it replaces the current *best*. On the other hand, if a mutated individual is better than current best individual, then it replaces the current *best*. A new iteration starts with a new population S^{t+1} composed by the current best individual, with the remaining $(N-1)$ individuals chosen randomly. We should remark that PSA is elitist: the best individual encountered is always kept in a population. The directive $last(S^t) \leftarrow best$ means that the best individual found up to generation t, denoted by *best*, is kept at the position that corresponds to the last individual of the population at step $t+1$.

Algorithm 10 Participatory Search with Arithmetical Recombination.

procedure PSAR
    f an objective function
    $s \in S^t$ and $s^{'} \in S^{t'}$
    set *best* randomly
    set $a_0 \leftarrow 0$; $t \leftarrow 0$
    while $t \leq t_{max}$ **do**
        generate population S^t randomly
        $last(S^t) \leftarrow best$
        $S^{t'} \leftarrow s' = argmax_{r \in S^t}(\rho(s, r))$
        find *best* in S^t
Selection:
        compute $\rho^s(s, best)$ and $\rho^{s^{'}}(s^{'}, best)$
        if $\rho^s \geq \rho^{s^{'}}$ **then**
            $p_{selected} \leftarrow s$
        else
            $p_{selected} \leftarrow s^{'}$
        end if
Recombination:
        choose $\alpha, \beta \in [0, 1]$ randomly
        compute $\rho_r = \rho(s, s^{'})$
        compute $a_{t+1} = a_t + \beta((1 - \rho_r) - a_t)$
        $p_r = (1 - \alpha\rho_r^{1-a_t})s + \alpha\rho_r^{1-a_t}s^{'}$
Mutation:
        compute $\rho_m = \rho(p_{selected}, p_r)$
        $p_m = best + \rho_m^{1-a_{t+1}}(p_{selected} - p_r)$
        if $f(p_r)$ better than $f(best)$ **then**
            $best \leftarrow p_r$
        end if
        if $f(p_m)$ better than $f(best)$ **then**
            $best \leftarrow p_m$
        end if
        $t \leftarrow t + 1$
    end while
    return *best*
end procedure

Algorithm 11 Participatory Search with Selective Transfer.

procedure PSST
    f an objective function
    $s \in S^t$ and $s^{'} \in S^{t'}$
    set *best* randomly
    set $a_0 \leftarrow 0$; $t \leftarrow 0$
    while $t \leq t_{max}$ **do**
        generate population S^t randomly
        $last(S^t) \leftarrow best$
        $S^{t'} \leftarrow s' = argmax_{r \in S^t}(\rho(s, r))$
        find *best* in S^t
Selection:
        compute $\rho^s(s, best)$ and $\rho^{s^{'}}(s^{'}, best)$
        if $\rho^s \geq \rho^{s^{'}}$ **then**
            $p_{selected} \leftarrow s$
        else
            $p_{selected} \leftarrow s^{'}$
        end if
Selective Transfer:
        choose h, k, $h \leq k$, and $r \in [0, 1]$ randomly
        if $r \leq 1/2$ **then**
            $c = [best(1 : h), p_{selected}(h + 1 : k), best(k + 1 : n)]$
        else
            $c = [p_{selected}(1 : h), best(h + 1 : k), p_{selected}(k + 1 : n)]$
        end if
Mutation:
        compute $\rho_m = \rho(p_{selected}, c)$
        $p_m = best + \rho_m^{1-a_{t+1}}(p_{selected} - c)$
        if $f(c)$ better than $f(best)$ **then**
            $best \leftarrow c$
        end if
        if $f(p_m)$ better than $f(best)$ **then**
            $best \leftarrow p_m$
        end if
        $t \leftarrow t + 1$
    end while
    return *best*
end procedure

Algorithm 12 Differential Participatory Search with Arithmetical Recombination.

procedure DPSA
    f an objective function
    $s \in S^t$ and $s^{'} \in S^{t'}$
    set *best* randomly
    set $a_0 \leftarrow 0$; $t \leftarrow 0$
    while $t \leq t_{max}$ **do**
        generate population S^t randomly
        $last(S^t) \leftarrow best$
        $S^{t'} \leftarrow s' = argmax_{r \in S^t}(\rho(s, r))$
        find *best* in S^t
Mutation:
        choose $\beta \in [0, 1]$ randomly
        compute $\rho_m = \rho(s, s^{'})$
        compute $a(t+1) = a(t) + \beta((1 - \rho_m) - a(t))$
        $p_m = best + \rho_m^{1-a_{t+1}}(s - s^{'})$
Recombination:
        choose $\alpha \in [0, 1]$ randomly
        compute $\rho_r = \rho(s, s^{'})$
        compute $a_{t+1} = a_t + \beta((1 - \rho_r) - a_t)$
        $p_r = (1 - \alpha\rho_r^{1-a_t})s + \alpha\rho_r^{1-a_t}s^{'}$
Selection:
        compute $\rho^{p_r}(p_r, best)$ and $\rho^{p_m}(p_m, best)$
        if $\rho^{p_r} \geq \rho^{p_m}$ **then**
            $p_{selected} \leftarrow p_r$
        else
            $p_{selected} \leftarrow p_m$
        end if
        if $f(p_{selected})$ better than $f(best)$ **then**
            $best \leftarrow p_{selected}$
        end if
        $t \leftarrow t + 1$
    end while
    return *best*
end procedure

Algorithm 13 Differential Participatory Search with Selective Transfer.

procedure DPST
    f an objective function
    $s \in S^t$ and $s^{'} \in S^{t'}$
    set *best* randomly
    set $a_0 \leftarrow 0$; $t \leftarrow 0$
    while $t \leq t_{max}$ **do**
        generate population S^t randomly
        $last(S^t) \leftarrow best$
        $S^{t'} \leftarrow s' = argmax_{r \in S^t}(\rho(s, r))$
        find *best* in S^t
Mutation:
        choose $\beta \in [0, 1]$ randomly; compute $\rho_m = \rho(s, s^{'})$
        compute $a(t+1) = a(t) + \beta((1 - \rho_m) - a(t))$
        $p_m = best + \rho_m^{1-a_{t+1}}(s - s^{'})$
Selective Transfer:
        choose h, k, $h \leq k$, and $r \in [0, 1]$ randomly
        if $r \leq 1/2$ **then**
            $c = [best(1 : h), p_m(h + 1 : k), best(k + 1 : n)]$
        else
            $c = [p_m(1 : h), best(h + 1 : k), p_m(k + 1 : n)]$
        end if
Selection:
        compute $\rho^c(c, best)$ and $\rho^{p_m}(p_m, best)$
        if $\rho^c \geq \rho^{p_m}$ **then**
            $p_{selected} \leftarrow c$
        else
            $p_{selected} \leftarrow p_m$
        end if
        if $f(p_{selected})$ better than $f(best)$ **then**
            $best \leftarrow p_{selected}$
        end if
        $t \leftarrow t + 1$
    end while
    return *best*
end procedure

We address the convergence analysis and the limiting behavior of PSAR using the random search theory [Solis and Wets, 1981]. The random search technique gives a way to model the behavior of the algorithms. The focus is on the convergence and the limiting behavior of the PSAR algorithm from the point of view of random search theory. The analysis of the remaining instances of PSA proceeds similarly because they fit the same conceptual framework as does PSAR. The following algorithm serves as a conceptual description of PSAR.

Algorithm 14 Conceptual PSAR algorithm.

```
1: procedure PSAR
2:     Set t ← 0
3:     Choose best^t
4:     repeat
5:         Generate S^t using μ_t
6:         Compute S^{t'}
7:         Find best^t in S^t
8:         Set last(S^t) = D(S^t, best^t)
9:         Set t ← t + 1
10:    until termination criterion is met
11: end procedure
```

where μ_t is a uniform probability distribution, and *best* is

$$best = argmin_{s_i}\{f(s_i)\}, s_i \in S^t. \tag{4.51}$$

Recall that PSAR, like all remaining instances of PSA, is elitist. Therefore, the best individual is always kept in the current population S^t. This is done via statement $last(S^t)$ which assigns *best* to the last position of the list of individuals that encodes S^t. The conceptual PSAR algorithm assumes that generation of new individuals using μ_t does not overwrites the last individual.

Function D, as stated in the conceptual PSAR algorithm, is as follows

$$D(S^t, best^t) = \begin{cases} p_r^t = (1 - \alpha\rho_r^t)s^t + \alpha\rho_r^t s^{t'} & \text{if } f(p_r^t) < f(best^t) \quad (4.52) \\ p_m^t = best^t + \rho_m^t(p_{selected}^t - p_r^t) & \text{if } f(p_m^t) < f(best^t) \quad (4.53) \\ best^t & \text{otherwise} \quad (4.54) \end{cases}$$

where p_m is as in (4.27), p_r as in (4.25) and $\rho_m, \rho_r, \alpha \in [0, 1]$.

We notice that if $f(p_r^t) < f(best^t)$, where p_r^t is as in (4.52), then p_r^t becomes current *best*. Otherwise, if $f(p_m^t) < f(best^t)$, where p_m^t is as in (4.53), then p_m^t becomes current *best*. Otherwise, $D(S^t, best^t) = best^t$ and the *best* found so far is kept. Hence, the sequence $\{f(s^t)\}_{t=0}^n$ is monotonic and nonincreasing.

Moreover, the sequence generated by PSAR produces the best solution in finite time, that is, the optimal solution will be found in finite number of steps and will never be lost. This is because the individual kept at $last(S^{t+1}) = D(S^t, best^t)$ is created by the successive application of the participatory selection, recombination and mutation steps. Further, because μ_t is an uniform distribution in $[0, 1]$, any subset $S^t \subset [0, 1]^{N\times n}$ has a

positive measure, which means zero probability of repeatedly missing a subset of S^t of $[0,1]^{N\times n}$. In other words, μ_t can not consistently ignore any part of $[0,1]^{N\times n}$.

We conclude that the sequence $\{f(s^t)\}_{t=0}^{n}$ produced by PSAR is monotonic and non-increasing, and thus satisfies Assumption 1 of Chapter 1. Also, recall that because μ_t as a uniform distribution in $[0,1]$ any $S^t \subset [0,1]^{N\times n}$ has a positive measure, and hence it satisfies Assumption 2 as well. Therefore we have the following result.

Theorem 5. *Suppose that f is a measurable function, S is a measurable subset of $[0,1]^{N\times n}$ and that Assumption 1 and Assumption 2 hold. Let $\{best^t\}_{t=0}^{\infty}$ be a sequence generated by the algorithm PSAR. Thus, PSAR converges with probability 1 to the global optimum.*

Proof. Expressions (4.52,4.53,4.54) generate monotonic sequences and S is contained in $[0,1]^{N\times n}$. Therefore, Assumption 1 and Assumption 2 hold and hence, from Theorem (1) of Chapter 1, (3.5) holds and the convergence of PSAR is proved.

□

4.5 Illustrative Example using PSAR

The participatory search process is illustrated using PSAR and a sphere function with two variables.

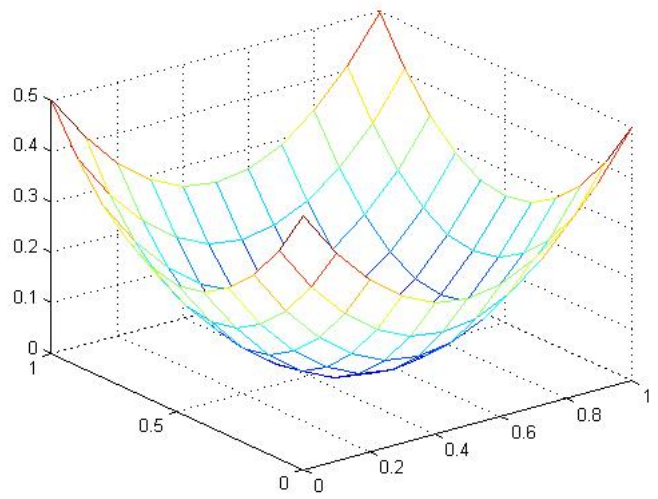


Figure 4.11: Sphere function.

The sphere function we consider in the simplest one

$$f(s_1, s_2) = (s_1 - 0.5)^2 + (s_2 - 0.5)^2 \tag{4.55}$$

where $s_1, s_2 \in [0,1]$. The problem is to find s^* that minimizes $f(s_1, s_2)$.

The sphere function is continuous, convex and unimodal function as Figure 4.11 shows. The global minimum at $s^* = (0.5, 0.5)$ and $f(s^*) = 0$.

Assume a population with $four$ individuals. Table 4.1 shows an initial population.

Table 4.1: Initial population.

Individual	(s_1, s_2)	$f(s_1, s_2)$
1	(0.0305 0.9047)	0.3842
2	(0.7441 0.6099)	0.0717
3	(0.5000 0.6177)	0.0139
4	(0.4799 0.8594)	0.1296

Table 4.2: Compatibility degrees for the 4 individuals.

$\rho(s1, s2)$	1	2	3	4
1		0.4232	0.5673	**0.7171**
2	0.4232		**0.8560**	0.7062
3	0.5673	**0.8560**		0.8502
4	0.7171	0.7062	**0.8502**	

From Table 4.1, individual 3 ($s_1 = 0.500$ and $s_2 = 0.6177$) is the current best and $f(s_1, s_2) = 0.0139$. To form the mating pool, the most compatible individual amongst the remaining individuals is found. This is repeated for each individual of the current population. For instance the compatibility degrees of individual 1, namely $\rho(1, 2), \rho(1, 3)$ and $\rho(1, 4)$, are

$$\rho(1,2) = 1 - \frac{1}{2}[0.7136 + 0.2948] = 0.4232,$$

$$\rho(1,3) = 1 - \frac{1}{2}[0.4695 + 0.287] = 0.5673,$$

$$\rho(1,4) = 1 - \frac{1}{2}[0.4494 + 0.0453] = 0.7171.$$

Table 4.2 shows the values of the compatibility degrees for the remaining individuals of the population.

The highest compatibility values, marked bold in the Table 4.2, define the most compatible individuals. Highest compatibility individuals assemble the pool shown in Table 4.3.

Next, the algorithm uses (4.19), (4.20), (4.21) and each pair of individuals of the mating pool of Table 4.3 to select the individuals which are closest to current best individual,

Table 4.3: Mating pool.

Individual (S)	Mating Pool (S')
1	4
2	3
3	2
4	3

Table 4.4: Selection, Recombination and Mutation.

$p_{selected}$	p_r	p_m	$f(p_m)$
4	(0.2777 0.7536)	(0.6851 0.5045)	0.0343
3	(0.7441 0.6099)	(0.2867 0.6245)	0.0609
3	(0.6853 0.6117)	(0.3339 0.6230)	0.0427
3	**(0.4883 0.7583)**	**(0.5107 0.4888)**	**0.0002**

shown in the first column of Table 4.4.

Recombination is done for each pair of corresponding individuals of the current mating pool using (4.25). The recombined individuals produced are in the second column of Table 4.4.

Mutation uses (4.27) the recombined individuals p_r, the selected individuals $p_{selected}$ with the current best individual $s^* = (0.500, 0.6177)$ to produce the mutant population with individuals p_m shown in the third column of Table 4.4, with the respective values of the objective function for each mutant individual collected in the fourth column.

We notice from the Table 4.4 that the value of the objective function for the mutant $p_m = (0.5107, 0.4888)$ is $f(p_m) = 0.0002$ (marked bold) which is better than $f(s^*) = 0.0139$ of the current best individual $s^* = (0.5000, 0.6177)$. Thus mutant $p_m = (0.5107, 0.4888)$ replaces current best in the population of the next generation.

The steps of the participatory search detailed above are illustrated in Figure 4.12. The figure shows the contour of the objective function, the initial population, and the best mutant and its respective generators p_r and $p_{selected}$. The orange dot is the optimal solution $s^* = (0.5, 0.5)$ we are looking for, the purple diamond is the mutation individual $p_m = (0.5107, 0.4888)$, the green triangle is the individual $p_r = (0.4883, 0.7583)$. The blue stars are the initial population.

Notice from Table 4.3, 4.4, and Figure 4.12 that the individual 4 has individual 3 as a mate, and that the individual 3 is selected as $p_{selected}$ in the selection step. Thus, the individuals 4 and 3 produce offspring p_r. Participatory mutation produces p_m using the combination of *best* with a variation whose amount is proportional to $(p_{selected} - p_r)$, but points in the direction from individual 3 to the mutant. Recall that the individual 3 is the current population *best* and that a new best individual is produced in this step. Thus, as Figure 4.12 suggests, search progress towards the optimal solution.

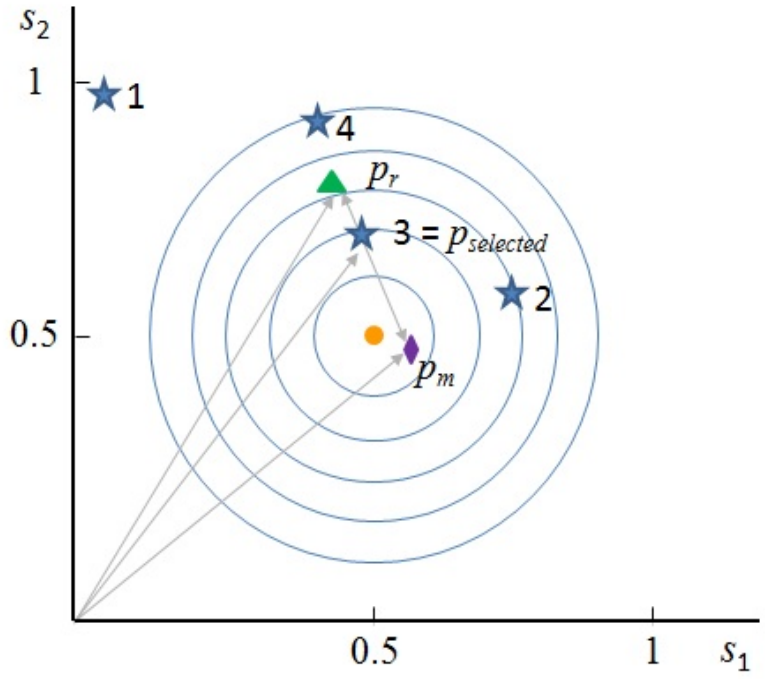


Figure 4.12: A step of PSAR.

4.6 Summary

This chapter has introduced the notion of participatory search, the search operators it uses, the nature of the operators. It has been shown that selective transfer and participatory recombination emphasize intensification, while participatory mutation enhances diversification strategy. We also addressed the convergence of the participatory search algorithms, focusing on PSAR, the participatory search with arithmetical recombination procedure. The notions of random search and convergence in probability have shown that the participatory search learning algorithm converges to global infimum. Next chapter evaluates the performance of PSAR using benchmark optimization problems available in the literature and compares PSAR with the current state of the art population-based optimization algorithms.

Chapter 5

Computational Results

This chapter investigates the performance of the participatory search algorithm, focusing in the PSAR. First we evaluate and compare PSAR with biogeography-based optimization (BBO), and genetic algorithm (GA) using classic benchmark, real-valued optimization problems of different dimension, and considering populations of different sizes. Next, the performance of PSAR is evaluated in front of BBO, GA, ant colony (ACO), differential evolution (DE), evolution strategy (ES), probability-based incremental learning (PBIL), particle swarm (PSO), and the stud genetic algorithm (SGA) using fixed population size and number of generations. PSAR is also evaluated against the winners of the 2013 IEEE Congress on Evolutionary Computation competition.

5.1 Evaluation of PSAR, BBO and GA algorithms

We start with an overview of the ten benchmark optimization problems to be considered during the evaluations. The benchmark functions, listed in Table 5.1, are amongst the most widely used in the literature to evaluate evolutionary optimization algorithms [Simon, 2008], [Simon et al., 2011]. The test problems concern finding the global minimum of each function $f_i(s), s \in \mathbb{R}^n, i = 1, ..., 10$, where n is the dimension of the decision space. Functions f_1, f_2 and f_4 are multimodal functions in which the number of local minimum increases exponentially. Function f_3 is a noisy quartic function, with $random[0, 1)$ a uniformly distributed random variable in $[0, 1)$. Functions f_5 to f_9 are unimodal functions. Function f_{10} is the step function which has one minimum and is discontinuous. Additional information about the defining characteristics of the benchmark functions are summarized in Table 5.1 and detailed in Appendix A. Recall that a function is separable if it can be expressed as a sum of functions of just one variable. A function is regular if it is analytic at each point of its domain. See [Simon, 2008] and [Yao and Liu, 1996] for further details.

First, we evaluate and compare PSAR with BBO and GA for $n = 5, 10, 20$ and 30 with the population size fixed at $N = 50$. We run PSAR, BBO, and GA 100 times for each benchmark function f_i with 200 generations. The results are shown in Tables 5.2, 5.3, 5.4 and 5.5. PSAR performance improves upon BBO and GA as dimension of the decision variable increases. For $n = 5$ BBO performs best on two of the ten benchmarks, whereas PSAR performs best in eight of them. As n increases BBO performs best for only one of the benchmarks while PSAR performs best for the remaining nine benchmarks. Also, the standard deviation of the PSAR is larger as the dimension of the decision variable increases, minus the function f_3, f_4, f_9 that are separable regular.

Table 5.1: Characteristics of the Benchmark Test Functions.

Function	Multimodal	Separable	Regular	Domain
f_1 Ackley	Yes	No	Yes	[-32, 32]
f_2 Griewank	Yes	No	Yes	[-600, 600]
f_3 Quartic	No	Yes	Yes	[-1.28, 1.28]
f_4 Rastrigin	Yes	Yes	Yes	[-5.12, 5.12]
f_5 Rosenbrock	No	No	Yes	[-30, 30]
f_6 Schwefel 1.2	No	No	Yes	[-100, 100]
f_7 Schwefel 2.21	No	No	No	[-100,100]
f_8 Schwefel 2.22	Yes	No	No	[-10, 10]
f_9 Sphere	No	Yes	Yes	[-100, 100]
f_{10} Step	No	Yes	No	[-100, 100]

As [Derrac et al., 2011] and [Antonelli et al., 2013] suggest, nonparametric tests for multiple comparison must be done to verify if there exist statistical differences among the performance of PSAR, BBO and GA. Here the null hypothesis H_0 is that all the algorithms are equivalent. Hence, rejection of this hypothesis means that there are differences in the performance of the algorithms studied. To accept H_0 means that the difference in performance is not statistically relevant. In other words, there is no statistical evidence to support similar performance of the algorithms.

First, the Friedman test is used to rank the algorithms. Friedman's test way of working is described as follows: It ranks the algorithms for each data set separately, the best performing algorithm getting the rank of 1, the second best rank 2, and so on. In case of ties average ranks are assigned. Let r_i^j be the rank of the j-th of k algorithms on the i-th of n data sets. The Friedman test compares the average ranks of algorithms,

$$R_j = \frac{1}{n}\sum_i r_i^j. \tag{5.1}$$

Under the null hypothesis, which states that all the algorithms are equivalent and so their ranks R_j should be equal, the Friedman statistic:

$$\chi_F^2 = \frac{12n}{k(k+1)}\left[\sum_j R_j^2 - \frac{k(k+1)^2}{4}\right] \tag{5.2}$$

is distrubuted according to χ_F^2 with $(k-1)$ degrees of freedom [Sheskin, 2006].

Next, the Iman and Davenport test evaluates whether there exist statistically relevant differences in the performance of the algorithms. In [Iman and Davenport, 1980], Iman and Daveport showed that friedmans χ_F^2 presents a conservative behavior and proposed a better statistic

$$F_F = \frac{(n-1)\chi_F^2}{n(k-1)\chi_F^2} \tag{5.3}$$

which is distributed according to the $F-$distribution with $(k-1)$ and $(k-1)(n-1)$ degrees of freedom.

If there is statistical difference, then the Holm's post-hoc procedure is used for further evaluation. Holm's test is a multiple comparison procedure that work with a control

algorithm (usually the one assumed to be the best). These are nonparametric tests, conducted using the R package `http://CRAN.R-project.org`.

Table 5.2, 5.3, 5.4 and 5.5 show the Friedman rank and Iman-Davenport p-values. Recall that, if the p-value is lower than the significance level ε ($\varepsilon = 0.05$ in this work), then the null hypothesis is rejected and we conclude that there are statistical differences between the performance of the algorithms. In other words, the algorithms do not have the same performance. Table 5.2, 5.3, 5.4 and 5.5 show that the p-values are lower than 0.05. There exist differences in the performance of the three algorithms. Thus, we proceed with the Holm post-hoc test with PSAR as the control algorithm. Holm's test rejects the hypothesis because $p < \varepsilon/i$, where i is the order of the significant p-value. Table 5.6 shows that the difference of the performance is not statistically relevant between PSAR and BBO because the null hypothesis is accepted, but the PSAR outperforms GA because PSAR is ranked higher than GA, and the hypothesis in the Holm' test is rejected.

Next we evaluate and compare PSAR with BBO and GA for population sizes $N = 50, 100$ and 200, keeping the dimension of the decision variable s fixed at $n = 30$. As before, we run PSAR, BBO and GA 100 times for each benchmark function. The results are shown in Tables 5.5, 5.7 and 5.8. The performance of PSAR improves upon BBO and GA as the population size increases. For $N = 50$, BBO performs best for only one of the benchmarks, while PSAR performs best for the remaining nine benchmarks. As the population size increases up to $N = 200$, BBO performs best for two of the benchmarks, and PSAR performs best for eight. On the other hand, the standard deviation of the PSAR is smaller as the size of population increase at 100. When the size of population is 200, the standard deviation of few function is greater. It means that there is a limit of population size to improve performance when the size of the population increases. Thus, the performance of the PSA is extremely sensitive to population size.

Table 5.5, 5.7 and 5.8 show that the p-values are lower than 0.05. Thus, we proceed with the Holm post-hoc test with PSAR as the control algorithm, Table 5.9. The hypothesis cannot be rejected for BBO, but it is rejected for GA.

Summing up, the ranking and nonparametric tests show that PSAR is significantly better than GA and BBO when looking at the rankings.

Tabela 5.2: Comparison of BBO, PSAR and GA with $n = 5$ and $N = 50$.

	BBO			PSAR			GA		
Function	Mean	SD	Best	Mean	SD	Best	Mean	SD	Best
f_1	2.9	1.2	0.036	**0.5556**	0.1409	0.0679	3.6	1.7	**0**
f_2	1.02	0.022	1.0003	**0.0472**	0.0182	**0**	1.037	0.027	1.0032
f_3	**6.2E-22**	6.2E-24	**0**	0.0144	0.0016	0.003	6.1E-7	1E-8	**0**
f_4	0.9	0.033	**0**	**0.3088**	0.3084	**0**	2	0.16	**0**
f_5	3.8	1.8	**0**	**0.5698**	0.2419	0.2637	7.9	2	**0**
f_6	150	16	0.34	**10.2512**	9.116	**0.0001**	290	61	2.8
f_7	4.1	1.7	0.26	**0.0668**	0.0269	**0.0128**	5	2.3	0.068
f_8	0.29	0.048	**0**	**0.0001**	0.0003	**0**	0.48	0.1	**0**
f_9	**0.0064**	0.00022	**0**	32.3582	6.7046	**0**	0.024	0.0018	**0**
f_{10}	15	2.5	**0**	**4.85**	4.1203	**0**	14	4.1	**0**
Friedman rank	2.1			1.3			2.6		
p-value	0.006351								
H_0	Rejected								

Tabela 5.3: Comparison of BBO, PSAR and GA with $n = 10$ and $N = 50$.

	BBO			PSAR			GA		
Function	Mean	SD	Best	Mean	SD	Best	Mean	SD	Best
f_1	4.1	2.6	0.75	**1.13**	0.9041	**0.705**	4.9	3.1	0.82
f_2	1.6	1.2	1	**0.9648**	0.4292	**0.6825**	1.8	1.3	1.016
f_3	**1.9E-7**	2E-9	**0**	0.1191	0.0204	0.031	0.00019	2.2E-6	**0**
f_4	3	0.28	**0**	**0.1543**	0.0646	0.0004	3	0.75	**0**
f_5	82	14	**0**	69.2734	36.58	0.1052	**74**	20	5.5
f_6	1200	380	55	**378.6823**	65.6126	**10.4554**	2800	790	89
f_7	15	8.2	3.8	**6.4992**	5.4036	**0.0077**	23	10	3.5
f_8	0.64	0.22	**0**	**0.2248**	0.0644	**0**	0.96	0.41	**0**
f_9	**0.11**	0.0077	**0**	0.2855	0.1914	0.0239	0.17	0.023	**0**
f_{10}	74	25	5	**18.4**	15.3747	**0**	140	38	8
Friedman rank	2.05			1.35			2.6		
p-value	0.01127								
H_0	Rejected								

Tabela 5.4: Comparison of BBO, PSAR and GA with $n = 20$ and $N = 50$.

	BBO			PSAR			GA		
Function	Mean	SD	Best	Mean	SD	Best	Mean	SD	Best
f_1	6	4.5	3.1	**1.877**	1.7893	**0.0131**	7.3	5.2	3
f_2	7.5	3.3	1.6	**5.9377**	3.2	**0.9999**	6.7	4	2.3
f_3	**0.0006**	0.00002	**0**	0.4373	0.4007	0.321	0.0026	0.0001	**0**
f_4	6.8	2.7	**0**	**5.4372**	0.7255	0.0031	10	3.8	0.013
f_5	150	68	**12**	**69.8631**	40.01	20.3841	190	96	18
f_6	6400	3100	1000	**5503.9696**	1533.4439	**147.3146**	9500	5300	1400
f_7	38	25	11	**4.4398**	4.2564	**0.0168**	41	31	17
f_8	3.6	1.8	0.22	**0.0025**	0.0015	**0**	5.8	2.6	0.96
f_9	1.3	0.3	**0.0061**	**0.6873**	0.6577	0.0377	1.9	0.63	0.012
f_{10}	430	250	82	**346**	135.6852	**0**	680	350	130
Friedman rank		2			1.2			2.8	
p-value	1.01E-4								
H_0	Rejected								

Tabela 5.5: Comparing BBO, PSAR and GA for $n = 30$ and $N = 50$.

	BBO			PSAR			GA		
Function	Mean	SD	Best	Mean	SD	Best	Mean	SD	Best
f_1	8	6.4	4.1	**2.4703**	1.8863	**0.0057**	9.3	7.1	4.7
f_2	16	9.2	4.5	**3.0606**	2.7007	**1**	19	12	5.1
f_3	**0.049**	0.0038	**7.7E-8**	0.2903	0.2808	0.1098	0.37	0.031	1E-5
f_4	14	7.4	2	**10.1666**	4.808	**0.0036**	16	9.7	5
f_5	340	170	38	**280.9419**	235.8698	**0.2259**	370	200	61
f_6	14000	8900	5000	**11029.08**	7032.9442	**186.4789**	27000	15000	6300
f_7	49	36	24	**3.0283**	2.9872	**0.0053**	58	43	30
f_8	10	5.9	1.6	**0.0004**	0.0002	**0**	13	7.4	3.8
f_9	4.2	1.9	0.41	**3.4653**	1.2266	**0.21**	6.6	2.8	0.85
f_{10}	1500	900	340	**1302.01**	504.309	**30**	2600	1200	650
Friedman rank		2.1			1			2.9	
p-value	3.87E-10								
H_0	Rejected								

Table 5.6: Holm's Post-hoc test for BBO, PSAR and GA with $\varepsilon = 0.05$ and different dimensions.

Dimension n=5				
Control algorithm: PSAR				
i	Algorithm	p-value	ε/i	H_0
2	GA	0.00365	0.025	Rejected
1	BBO	0.26355	0.05	Accepted
Dimension n=10				
Control algorithm: PSAR				
i	Algorithm	p-value	ε/i	H_0
2	GA	0.00518	0.025	Rejected
1	BBO	0.21875	0.05	Accepted
Dimension n=20				
Control algorithm: PSAR				
i	Algorithm	p-value	ε/i	H_0
2	GA	0.00034	0.025	Rejected
1	BBO	0.07363	0.05	Accepted
Dimension n=30				
Control algorithm: PSAR				
i	Algorithm	p-value	ε/i	H_0
2	GA	0.00002	0.025	Rejected
1	BBO	0.07363	0.05	Accepted

Figure 5.1, 5.2, 5.3, 5.4, 5.5 and 5.6 show how PSAR converges using the Griewank function with different dimension and the size of population as illustrations. The variation of the best fitness (lower curve) and the mean fitness in each generation. We can see that in the begining there is a strong decline in both lines, showing that the bad solutions are eliminated quickly and the best solutions are obtained with each new generation. The downward trend is declining over the generations, seeming to be nearly stabilized around 100 generations, but even after 200 generations we can still notice a small decrease in Figure 5.1 and 5.6.

Figure 5.4, 5.5 and 5.6 show how PSAR converges using the Griewank function with the different size of population as illustrations.

Tabela 5.7: Comparison of BBO, PSAR and GA with $n = 30$ and $N = 100$.

	BBO			PSAR			GA		
Function	Mean	SD	Best	Mean	SD	Best	Mean	SD	Best
f_1	5.548	0.592	3.565	**3.008**	2.649	**0.034**	17.521	1.023	14.619
f_2	4.179	0.915	2.333	**3.924**	0.908	**1**	48.297	18.998	14.71
f_3	**0.017**	0.013	**0.002**	0.338	0.148	0.248	1.72	1.121	0.194
f_4	24.718	4.568	14.988	**23.305**	3.392	**3E-4**	256.892	31.799	189.037
f_5	**103.569**	28.074	**41.179**	276.174	203.134	90.96	809.414	292.43	347.062
f_6	470.933	143.051	**213.314**	**208.927**	46.835	245.112	1799.376	565.896	779.148
f_7	7187.019	2102.306	3662.482	**3.242**	3.238	**0.005**	13970.008	3545.413	4426.331
f_8	4.482	1.066	2.1	**0.013**	0.002	**0**	54.443	9.177	36.7
f_9	0.969	0.337	0.372	**0.348**	0.187	**0.161**	39.521	11.795	15.44
f_{10}	357	124.738	126	**147**	93.367	**0**	6242.19	2603.085	1814
Friedman rank	2.2			1			2.8		
p-value	6.87E-8								
H_0	Rejected								

Tabela 5.8: Comparing BBO, PSAR and GA for $n = 30$ and $N = 200$.

	BBO			PSAR			GA		
Function	Mean	SD	Best	Mean	SD	Best	Mean	SD	Best
f_1	4.005	0.468	2.84	**2.869**	0.4794	**0.114**	17.488	1.046	14.031
f_2	2.187	0.379	1.464	**2.008**	1.381	**0.15**	33.843	15.53	8.461
f_3	**0.004**	0.005	**2E-4**	0.283	0.222	0.067	1.155	0.758	0.205
f_4	12.661	2.61	6.065	**11.916**	9.833	**0.006**	250.508	25.376	174.029
f_5	**71.349**	30.591	33.102	75.583	5.408	**19.602**	690.155	249.876	271.77
f_6	175.069	57.903	**75.44**	**142.733**	53.832	93.688	1167.761	371.306	306.297
f_7	4093.147	1372.64	1326.526	**2.659**	2.551	**0.085**	12182.611	3550.315	5231.045
f_8	1.915	0.541	0.9	**0.028**	0.0158	**0**	49.208	8.237	27
f_9	0.333	0.093	0.181	**0.212**	0.181	**0.001**	39.031	10.606	13.354
f_{10}	130.27	46.89	54	**125.1**	117.712	**0**	3712.85	1619.69	897
Friedman rank	2.2			1			2.8		
p-value	6.87E-8								
H_0	Rejected								

Table 5.9: Holm's Post-hoc test for BBO, PSAR and GA for $\varepsilon = 0.05$ and different population sizes.

Population size 50				
Control algorithm: PSAR				
i	Algorithm	p-value	ε/i	H_0
2	GA	0.00002	0.025	Rejected
1	BBO	0.07363	0.05	Accepted
Population size 100				
Control algorithm: PSAR				
i	Algorithm	p-value	ε/i	H_0
2	GA	0.000069	0.025	Rejected
1	BBO	0.179712	0.05	Accepted
Population size 200				
Control algorithm: PSAR				
i	Algorithm	p-value	ε/i	H_0
2	GA	0.00729	0.025	Rejected
1	BBO	0.179712	0.05	Accepted

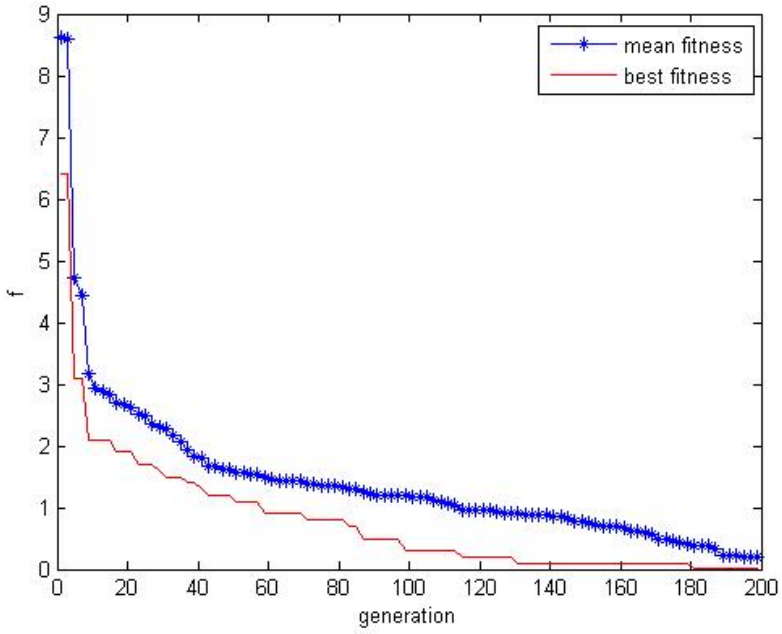


Figure 5.1: Convergence PSAR using the Griewank function with $n = 5$ and $N = 50$.

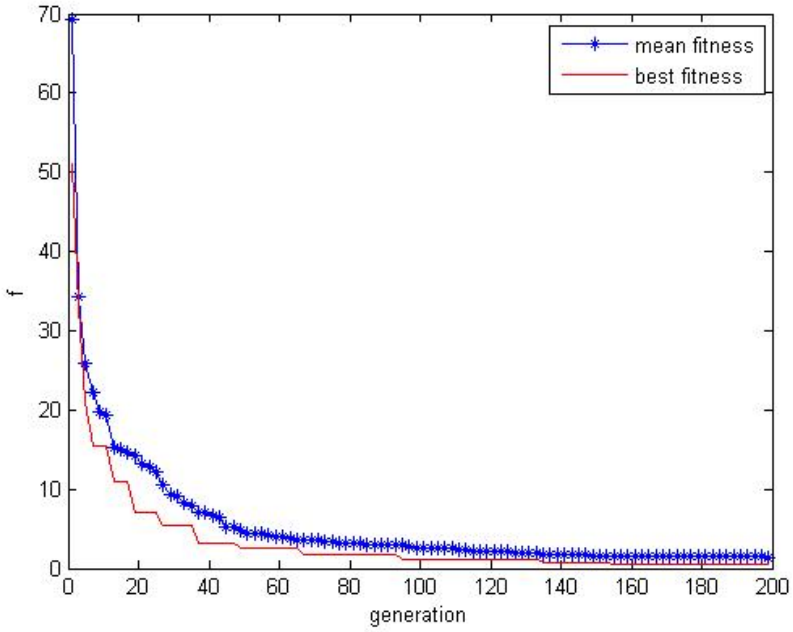


Figure 5.2: Convergence PSAR using the Griewank function with $n = 10$ and $N = 50$.

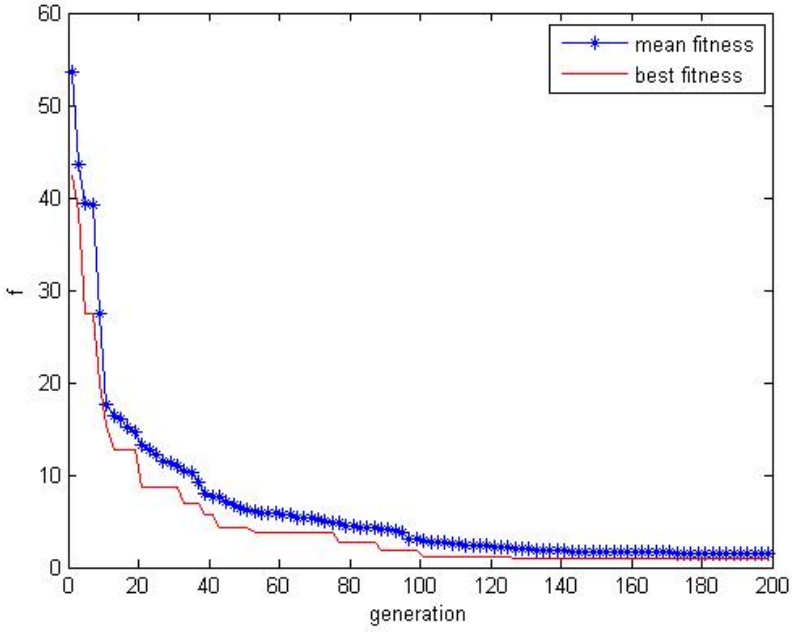


Figure 5.3: Convergence PSAR using the Griewank function with $n = 20$ and $N = 50$.

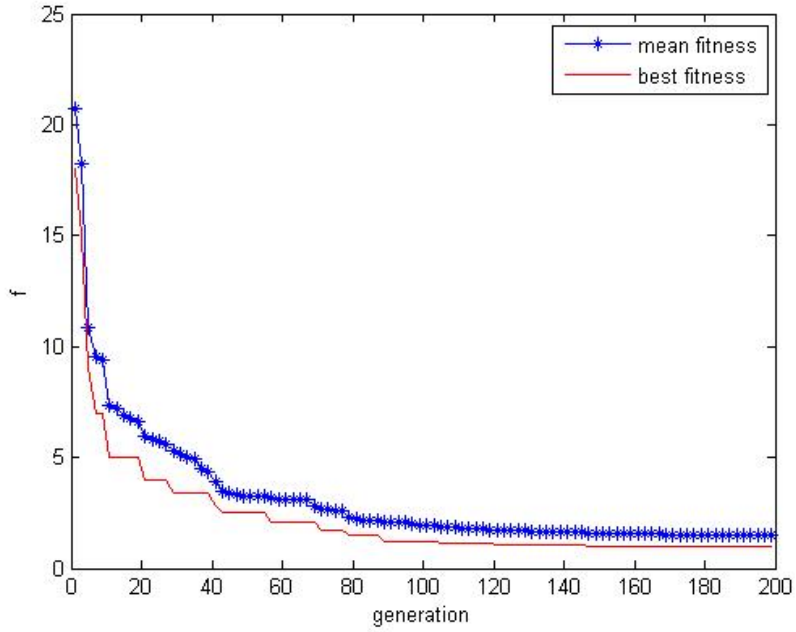


Figure 5.4: Convergence PSAR using the Griewank function with $n = 30$ and $N = 50$.

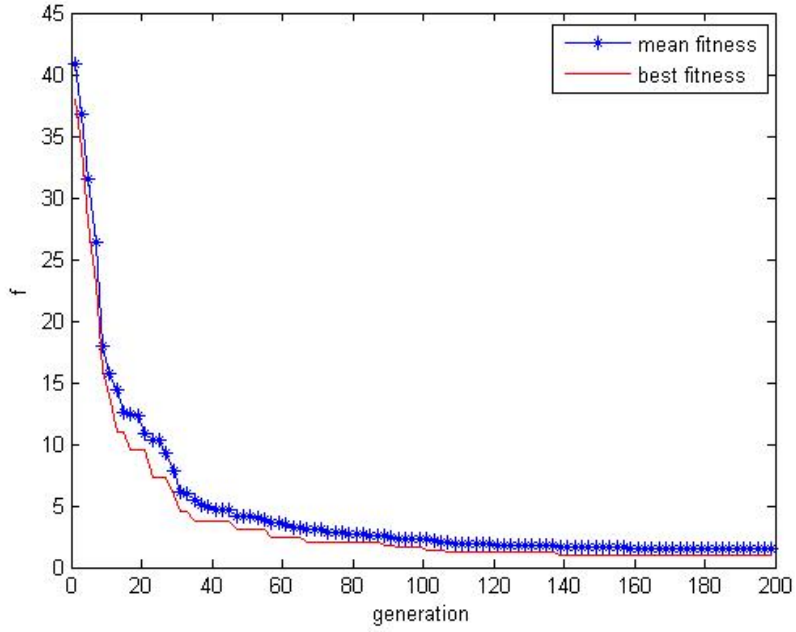


Figure 5.5: Convergence PSAR using the Griewank function with $n = 30$ and $N = 100$.

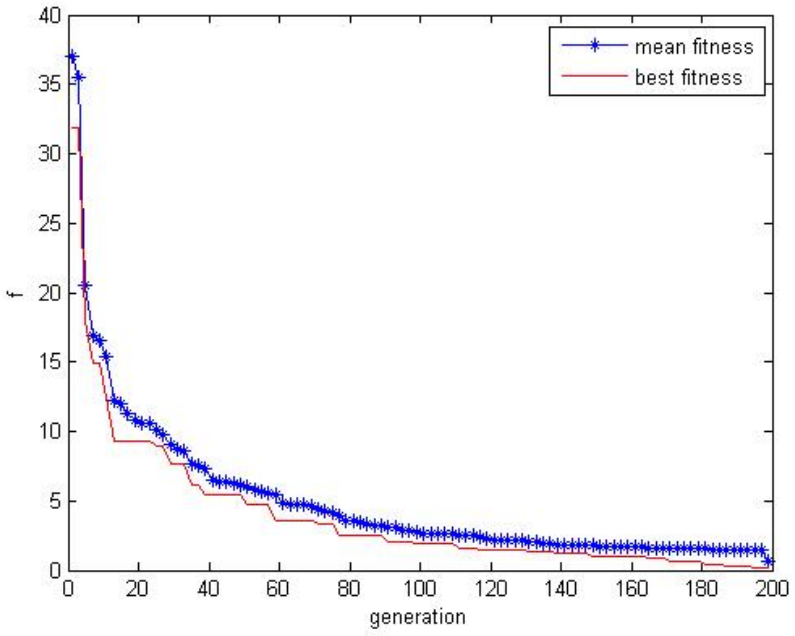


Figure 5.6: Convergence PSAR using the Griewank function with $n = 30$ and $N = 200$.

5.2 Evaluation of PSAR, ACO, BBO, DE, ES, GA, PBIL PSO and SGA algorithms

This section evaluates and compares PSAR with state of the art population-based algorithms using the real-valued benchmark optimization problems of [Simon, 2008] and [Yao and Liu, 1996], the ones adopted in the previous section. See Table 5.1.

The parameters used by the population-based optimization algorithms considered here are the same as the ones reported in [Simon, 2008]. They are:

1. ACO initial pheromone value $\tau_0 = 1E - 6$, pheromone update constant $Q = 20$, exploration constant $q_o = 1$, global pheromone decay rate $\rho_g = 0.9$, local pheromone decay rate $\rho_l = 0.5$, pheromone sensitivity $\alpha = 1$, and visibility sensitivity $\beta = 5$.

2. BBO habitat modification probability= 1, immigration probability bounds per gene= $[0, 1]$, step size for numerical integration of probabilities = 1, maximum immigration and migration rates for each island = 1, and mutation probability = 0.

3. DE with differential $F = 0.5$ and crossover constant $CR = 0.5$.

4. ES $\lambda = 10$ offspring each generation, and standard deviation $\sigma = 1$.

5. GA roulette wheel selection, single point crossover with crossover probability = 1, and mutation probability = 0.01.

6. PBIL learning rate = 0.05, good population member = 1 and 0 bad population member = 0, elitism parameter = 1, and probability vector mutation rate = 0.

7. PSO global learning, inertial constant = 0.3, cognitive constant = 1, and social constant for swarm interaction = 1.

8. SGA single point crossover with crossover probability = 1 and mutation probability = 0.01.

The results, summarized in Table 5.10, were produced using the computer codes available at `http://academic.csuohio.edu/simond/bbo` for ACO, BBO, DE, ES, GA, PBIL, PSO and SGA. The decision variable is $n = 20$ dimensional. Each algorithm uses a population of size 50 and are run for 200 generations. The results are the average of 100 runs of each algorithm. Table 5.10 highlights the following. Strictly speaking, PSAR is the most effective because it gives the best value of the objective function for five, out of ten functions. Also, PSAR, SGA, DE and BBO perform better than remaining algorithms because they achieve rank of $1.7, 2, 3.5$ and 4.6 in the Friedman test, respectively, with the p-values lower than 0.05, Table 5.11. From the Holm post-hoc procedure, with PSAR as control algorithm, Table 5.12, we can see that the hypothesis cannot be rejected for SGA, DE and BBO. Thus, PSAR outperforms the previous approaches, and that the performance difference between SGA, DE and BBO is not statistically relevant. In particular, the rank of PSAR is higher than the remaining algorithms.

Overall, from the results of this and Section 5.1 we conclude that PSAR performs best amongst ACO, BBO, DE, ES, GA, PBIL PSO and SGA.

Function		ACO	BBO	DE	PSAR	ES	GA	PBIL	PSO	SGA
f_1	Mean	13.121	6	5.58	**1.789**	18.898	7.3	19.532	16.131	5.325
	SD	1.475	4.5	0.55	0.877	0.517	5.2	**0.296**	0.619	0.835
	Best	10.385	3.1	4.302	**0.013**	17.287	3	18.59	14.294	3.449
f_2	Mean	6.603	7.5	2.383	9.003	104.6	6.7	219.491	73.401	**2.323**
	SD	2.335	3.3	**0.442**	5.937	24.744	4	29.349	12.081	0.676
	Best	2.686	1.6	1.534	**0.999**	49.779	2.3	148.283	42.608	1.339
f_3	Mean	0.283	**6E-4**	0.007	0.437	15.053	0.002	16.014	2.851	0.001
	SD	0.198	**2E-5**	0.004	0.4	4.336	1E-4	4.365	1.226	0.001
	Best	0.047	**0**	0.001	0.321	4.83	**0**	4.643	0.577	6.22E-5
f_4	Mean	130.794	6.8	111.335	**5.437**	216.551	10	219.219	155.856	24.502
	SD	17.707	**2.7**	10.124	1.7255	17.679	3.8	12.345	11.578	5.113
	Best	94.783	**0**	81.787	0.003	148.318	0.013	187.122	125.879	13.801
f_5	Mean	1638.838	150	83.114	**69.863**	2455.094	190	1846.62	542.895	72.436
	SD	461.459	68	**22.262**	40.01	569.146	96	343.952	176.688	29.905
	Best	744.811	**12**	37.055	20.384	830.463	18	1075.813	228.797	13.584
f_6	Mean	973.143	6400	2554.381	5503.969	3761.797	9500	5039.527	4718.627	**324.988**
	SD	258.235	3100	260.821	1533.443	326.785	5300	371.546	583.581	**140.444**
	Best	399.451	1000	2030.463	147.314	2909.344	1400	4043.034	3009.151	**96.639**
f_7	Mean	6317.128	38	8960.712	**4.439**	12248.32	41	12222.978	8051.164	5471.211
	SD	1807.57	25	1598.457	**4.256**	2256.061	31	1737.911	1858.519	1578.138
	Best	2624.572	11	4990.6	**0.016**	7520.481	17	7501.599	4201.678	2074.698
f_8	Mean	42.484	3.6	5.038	**0.015**	79.247	5.8	59.846	38.347	4.491
	SD	7.413	1.8	0.837	**0.002**	13.548	2.6	5.144	15.588	0.987
	Best	16.8	0.22	3.104	**0**	36.233	0.96	45.9	25.635	2.2
f_9	Mean	19.625	1.3	**0.415**	2.657	67.302	1.9	65.429	21.603	0.529
	SD	6.263	0.3	0.12	0.687	9.491	0.63	7.11	3.256	**0.224**
	Best	9.352	**0.006**	0.197	0.037	43.954	0.012	51.098	13.38	0.141
f_{10}	Mean	799.61	430	163.37	346	15692.78	680	24077.93	7978.4	**137.1**
	SD	285.842	250	**48.02**	135.685	2760.347	350	3414.351	1429.256	66.308
	Best	384	82	55	**0**	8466	130	15564	4952	27

Tabela 5.10: Optimal Values of the Benchmark Functions.

Table 5.11: Average Ranking of the Algorithms.

Algorithm	Friedman rank	p-value	H_0
ACO	6.3		
BBO	4.6		
DE	3.5		
PSAR	1.7		
ES	7.3	3.04E-14	Rejected
GA	5.2		
PBIL	7.8		
PSO	6.6		
SGA	2		

Table 5.12: Holm's Post-Hoc Test for $\varepsilon = 0.05$.

Control algorithm: PSAR				
i	Algorithm	p-value	ε/i	H_0
8	PBIL	4.82E-6	0.0062	Rejected
7	ES	1.5E-5	0.0071	Rejected
6	PSO	0.0019	0.0083	Rejected
5	ACO	0.00274	0.01	Rejected
4	GA	0.00864	0.0125	Rejected
3	BBO	0.4142	0.0166	Accepted
2	DE	0.5676	0.025	Accepted
1	SGA	0.683	0.05	Accepted

5.3 PSAR and the IEEE CEC 2013 competition

This section evaluates the PSAR against the winners of the IEEE CEC 2013 competition. The IEEE CEC 2013 competition was held as a Special Session on Real-Parameter Optimization of the IEEE Congress on Evolutionary Computation held on 2013. The test suite considers 28 benchmark functions involving a large number of features that arise in real world situations. The main characteristics of each of these functions is given in Table 5.13. Detailed description of the competition and the test suite is in the Appendix B [Liang et al., 2013a].

The PSAR algorithm was run for the 28 benchmark functions using a population with 200 individuals to compare with the results of the top two winning IEEE CEC 2013 algorithms, respectively ICMAESILS and NBIPOPaCMA [Liang et al., 2013b]. The computational experiments were performed according to the same experimental set-up of the IEEE CEC 2013 competition [Liang et al., 2013a]. It is summarized below:

1. Dimensions: $n = 10, 30$
2. Runs: 51
3. MaxFES: 10000*n (maximum number of function evaluations)
4. Search Range: $[-100, 100]^n$

Table 5.13: IEEE CEC 2013 competition test suite.

Characteristics	i	Functions	$f_i^* = f_i(s^*)$
Unimodal	1	Sphere Function	-1400
	2	Rotated High Conditioned Elliptic Function	-1300
	3	Rotated Bent Cigar Function	-1200
	4	Rotated Discus Function	-1100
	5	Different Powers Function	-1000
Basic Multimodal	6	Rotated Rosenbrock's Function	-900
	7	Rotated Schaffers Function	-800
	8	Rotated Ackley's Function	-700
	9	Rotated Weierstrass Function	-600
	10	Rotated Griewank's Function	-500
	11	Rastrigin's Function	-400
	12	Rotated Rastrigin's Function	-300
	13	Non-Continuous Rotated Rastrigin's Function	-200
	14	Schwefel's Function	-100
	15	Rotated Schwefel's Function	100
	16	Rotated Kasuura Function	200
	17	Lunacek Bi-Rastrigin Function	300
	18	Rotated Lunacek Bi-Rastrigin Function	400
	19	Expanded Griewank's plus Rosenbrock's Function	500
	20	Expanded Scaffer's Function	600
Composition	21	Composition Function 1 (n=5, Rotated)	700
	22	Composition Function 2 (n=3, Unrotated)	800
	23	Composition Function 3 (n=3, Rotated)	900
	24	Composition Function 4 (n=3, Rotated)	1000
	25	Composition Function 5 (n=3, Rotated)	1100
	26	Composition Function 6 (n=5, Rotated)	1200
	27	Composition Function 7 (n=5, Rotated)	1300
	28	Composition Function 8 (n=5, Rotated)	1400

5. Initialization: uniform random within the search space.

6. Global Optimization: all problems have the global optimum within the given bounds $f_i(s^*) = f_i(o) = f_i^*$, where o is the shifted global optimum.

7. Termination: terminate when either MaxFES or error smaller than 10^{-8} holds.

Table 5.14 shows the average error values produced by PSAR and winners ICMAESILS and NBIPOPaCMA [Liang et al., 2013b]. The average error value corresponds to the error measured at the maximum number of function evaluations. Average error values are reported for $n = 10$ and $n = 30$ in Table 5.14.

Table 5.14 shows that PSAR performs best in 15 ($n = 10$) and 16 ($n = 30$) of the benchmark functions while ICMAESILS performs best in 12 ($n = 10$) and 14 ($n = 30$), and NBIPOPaCMA performs best for 15 ($n = 10$) and 13 ($n = 30$) of the benchmark functions.

The results of the Friedman, and Iman and Davenport tests in Table 5.15 highlights the following: for $n = 10$ the Friedman test ranks PSAR = 1.9642, ICMAESILS = 2.1071 and NBIPOPaCMA = 1.9285. The large p-value= 0.0784 indicates that the difference of of performance of the algorithms is not statistically significant. This means that we do not have the statistical evidence to show the same performance between the algorithms in spite of the rank of NBIPOPaCMA be better than others. But, for $n = 30$, the p-value is lower than 0.05. Thus, there are differences in the performance of the three algorithms. Considering PSAR as the control algorithm in the Holm post-hoc test, we notice that the null hypothesis cannot be rejected for ICMAESILS, but it is rejected for NBIPOPaCMA. Hence PSAR outperforms NBIPOPaCMA, but does not outperform ICMAESILS.

Overall, although acceptance of the null hypothesis for $n = 10$ indicates that the difference of performance of algorithms is not statistically relevant, for $n = 30$ the Holm's test shows that the difference of performance between PSAR and NBIPOPaCMA is significant, and that the rank of PSAR is higher than NBIPOPaCMA. Thus, PSAR outperforms NBIPOPaCMA. On the other hand, the acceptance of null hypothesis in Holm's test for ICMAESILS means that we can not compare the performance of PSAR with ICMAESILS because the statistical evidence is insufficient, but the rank of PSAR is the highest. Thus, we conclude that PSAR can be considered as competitive with NBIPOPaCMA and ICMAESILS.

	$n = 10$			$n = 30$		
Function	PSAR	ICMAESILS	NBIPOPaCMA	PSAR	ICMAESILS	NBIPOPaCMA
f_1	**1.00E-8**	**1.00E-8**	**1.00E-8**	**1.00E-8**	**1.00E-8**	**1.00E-8**
f_2	**1.00E-8**	**1.00E-8**	**1.00E-8**	**1.00E-8**	**1.00E-8**	**1.00E-8**
f_3	**1.00E-8**	**1.00E-8**	**1.00E-8**	**1.00E-8**	**1.00E-8**	**1.00E-8**
f_4	**1.00E-8**	**1.00E-8**	**1.00E-8**	**1.00E-8**	**1.00E-8**	**1.00E-8**
f_5	**1.00E-8**	**1.00E-8**	**1.00E-8**	**1.00E-8**	**1.00E-8**	**1.00E-8**
f_6	**1.00E-8**	3.89E+0	**1.00E-8**	**1.00E-8**	**1.00E-8**	**1.00E-8**
f_7	**1.00E-8**	4.91E-6	**1.00E-8**	**1.00E-8**	7.01E-2	2.31E+0
f_8	6.66E+1	2.04E+1	**2.03E+1**	7.97E+1	**2.09E+1**	**2.09E+1**
f_9	5.71E+1	2.86E-1	**2.32E-1**	6.68E+1	4.34E+0	**3.30E+0**
f_{10}	**1.00E-8**	**1.00E-8**	**1.00E-8**	**1.00E-8**	**1.00E-8**	**1.00E-8**
f_{11}	**1.00E-8**	4.77E-1	3.64E-1	**1.00E-8**	2.25E+0	3.04E+0
f_{12}	**1.00E-8**	2.34E-1	2.38E-1	**1.00E-8**	1.72E+0	2.91E+0
f_{13}	**2.53E-1**	3.33E-1	4.84E-1	**1.00E-8**	2.16E+0	2.78E+0
f_{14}	**3.26E+1**	5.08E+1	1.15E+2	7.09E+2	**7.08E+2**	8.10E+2
f_{15}	1.52E+2	**4.42E+1**	1.58E+2	**1.52E+2**	2.59E+2	7.65E+2
f_{16}	3.94E+1	3.73E-1	**1.20E-1**	1.52E+2	**3.75E-1**	4.40E-1
f_{17}	6.52E+1	**1.12E+1**	1.13E+1	9.37E+1	**3.43E+1**	3.44E+1
f_{18}	8.61E+1	**1.12E+1**	1.13E+1	1.10E+2	**4.01E+1**	6.23E+1
f_{19}	9.83E+1	6.98E-1	**5.25E-1**	2.63E+2	2.24E+0	**2.23E+0**
f_{20}	1.17E+2	**2.72E+0**	2.73E+0	1.18E+2	1.44E+1	**1.29E+1**
f_{21}	**1.42E+2**	2.18E+2	1.53E+2	**1.76E+2**	1.88E+2	1.92E+2
f_{22}	4.72E+2	**1.66E+2**	1.75E+2	7.77E+2	**5.33E+2**	8.38E+2
f_{23}	2.05E+2	**4.08E+1**	1.74E+2	3.24E+2	**2.69E+2**	6.67E+2
f_{24}	1.59E+2	1.32E+2	**1.20E+2**	1.68E+2	2.00E+2	**1.62E+2**
f_{25}	1.92E+2	1.92E+2	**1.77E+2**	**2.08E+2**	2.40E+2	2.20E+2
f_{26}	1.58E+2	1.18E+2	**1.11E+2**	1.64E+2	2.16E+2	**1.58E+2**
f_{27}	**2.29E+2**	3.25E+2	3.17E+2	**2.63E+2**	3.00E+2	4.69E+2
f_{28}	**1.40E+2**	2.24E+2	2.49E+2	**1.96E+2**	2.45E+2	2.69E+2
Friedman rank	1.9642	2.1071	1.9285	1.6785	2	2.3214
p-value		0.0784			0.0426	
H_0		accepted			rejected	

Tabela 5.14: Average error values for PSAR, ICMAESILS and NBIPOPaCMA.

Table 5.15: Holm's Post-hoc test for PSAR, ICMAESILS and NBIPOPaCMA with $\varepsilon = 0.05$ for $D = 30$.

Dimension of 30				
Control algorithm: PSAR				
i	Algorithm	p-value	ε/i	H_0
2	NBIPOPaCMA	0.01615	0.025	rejected
1	ICMAESILS	0.2291	0.05	accepted

5.4 Summary

This chapter has evaluated and compared the performance of the participatory search algorithm PSAR with state of the art population-based algorithms. Computational results suggest that participatory search algorithm PSAR outperforms the previous approaches from the point of view of the quality of the solution.

The difference of the ranking of PSAR and the top two algorithms ICMAESILS and NBIPOPaCMA of IEEE CEC 2013 is small in low dimensional problem instances. PSAR, ICMAESILS, and NBIPOPaCMA perform differently in higher dimensional instances. Holm's test shows that the difference in performance between PSAR and NBIPOPaCMA is significant, and the rank of PSAR is higher than NBIPOPaCMA. On the other hand, the null hypothesis is accepted for ICMAESILS, which means that the performance between PSAR and ICMAESILS are not comparable due to lack of statistical evidence, but PSAR achieves higher rank.

Chapter 6

Participatory Search Algorithms in Fuzzy Modeling

This chapter addresses applications of participatory search algorithms in fuzzy rule-based system modeling. The aim is to illustrate potential applications of PSA and to evaluate and compare the performance of PSST, PSAR, DPST and DPSA algorithms using actual data and results reported in the literature.

6.1 Linguistic Fuzzy Models

The problem of interest here is to develop linguistic fuzzy models using actual data of electric maintenance (ELE), Auto MPG6 (MPG6), categorical data analysis (ANA), and abalone age (ABA). The features of the data sets are summarized in Table 6.1. The ELE, MPG6, ANA and ABA datasets are part of KEEL, an open state of the art genetic fuzzy system (GFS) environment [Alcalá-Fdez et al., 2009]. These data were adopted in [Alcalá et al., 2011], one of the most representative GFS reported in the literature [Fazzolari et al., 2013], the reason why it is adopted here for evaluation and comparison purposes. The same representation and encoding schemes of [Alcalá et al., 2011] are used by all PSA algorithms. They are summarized as follows.

1. Database encoding: $(C = C_1, C_2)$ a double-encoding scheme.
   First, equidistant strong fuzzy partitions are identified considering the granularity (labels) specified in C_1. Second, the membership functions of each variable are uniformly rearranged to a new position considering lateral displacement values specified in C_2.

   - Number of labels C_1: this is a vector of integers of size n representing the number of linguistic variables.

     $$C_1 = (L^1, ..., L^n). \tag{6.1}$$

     Gene L^i is the number of labels of the ith linguistic variable, $L^i \in \{2, ..., 7\}$.
   - Lateral displacements C_2: this is a vector of real numbers of size n that encodes displacements α^i of the different variables, $\alpha^i \in [-0.1, 0.1]$. A detailed description of the linguistic 2-tuple representation is given in [Herrera and Martínez, 2000] and [Alcalá et al., 2007].

     $$C_2 = (\alpha^1, ..., \alpha^n). \tag{6.2}$$

An example of the encoding scheme is given in Figure 6.1.

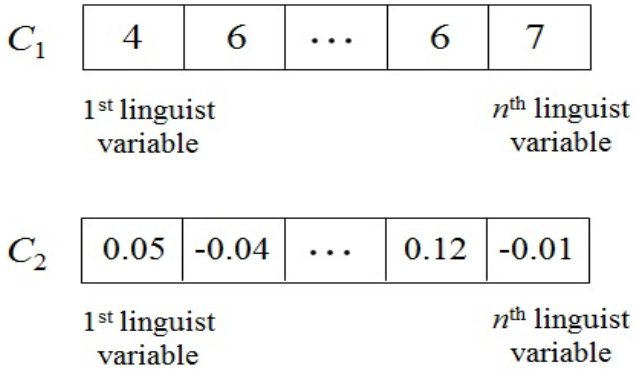


Figure 6.1: A double-encoding scheme C_1 and C_2.

Figure 6.2 illustrates the lateral displacement of V for $\alpha = -0.05$.

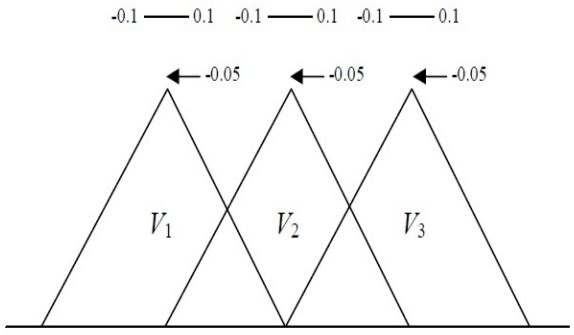


Figure 6.2: Lateral displacement of the linguistic variable V values V_1, V_2, and V_3.

2. Rule base: constructed using the Wang and Mendel algorithm (WM) [Wang and Mendel, 1992] and [Wang, 1994] as follows:

   (a) granulate the input and output spaces;

   (b) generate fuzzy rules using the given data;

   (c) assign a certainty degree to each rule generated to resolve conflicts;

   (d) create a fuzzy rule base combining the rules generated and rules provided by experts (if available);

   (e) determine the input-output mapping using the combined fuzzy rule base and a defuzzification procedure.

   An example of a fuzzy rule-base developed for ELE is shown in Figure 6.3.

   Example of rules of the rule base of Figure 6.3 include:

   rule 1: IF X1 is 1 and X2 is 1 and x3 is 1 and x4 is 1 THEN Y is 1

 # Rule base : 28

X1	X2	X3	X4	Y
1	1	1	1	1
2	1	1	2	3
3	3	2	3	4
3	2	1	3	3
3	4	3	2	4
1	2	1	1	1
1	1	1	2	2
1	2	1	2	2
2	2	2	3	3
4	2	1	2	3
..	..	..	..	..

Figure 6.3: Rule base constructed using WM algorithm.

rule 2: IF X1 is 2 and X2 is 1 and x3 is 1 and x4 is 2 THEN Y is 3

rule 3: IF X1 is 3 and X2 is 3 and x3 is 2 and x4 is 3 THEN Y is 4

⋮ ⋮ ⋮ ⋮

3. Objective function: the mean-squared error (MSE)

$$MSE = \frac{1}{2|D|}\sum_{l=1}^{|D|}(F(x^l) - y^l)^2 \tag{6.3}$$

where $|D|$ is the size of the dataset, $F(x)$ is the output of the FRBS model, and y the actual value of the output. Fuzzy inference uses the max-min procedure with center of gravity deffuzification.

4. Initial population: each chromosome has the same number of linguistic labels, from two to seven labels for each input variable. For each label of the inputs, all possible combinations are assigned to the respective rules consequents. Moreover, for each combination, two copies are added with different values in the C_2 part. The first has values randomly chosen in $[-0.1, 0]$ and the second random values chosen in $[0, 0.1]$.

5. Recombination: $p_r \leftarrow floor(p_r)$ for C_1.
If a gene g of p_r in C_1 is lower than 2, then $L_g = 2$, else if a gene g is higher than 7, then $L_g = 7$.

6. Mutation: $p_m \leftarrow floor(p_m)$ for C_1.
If a gene g of p_m in C_1 is lower than 2, then $L_g = 2$, else if a gene g is higher than 7, then $L_g = 7$.

The electric maintenance model has four input variables and one output variable. The ELE dataset contains electrical maintenance data and has 1056 samples. This is an

Table 6.1: Summary of the Datasets.

Problem	**Abbr.**	**Variables**	**Samples**
Electrical maintenance	ELE	4	1056
Auto MPG6	MPG6	5	398
Analact	ANA	7	4052
Abalone	ABA	8	4177

instance in which we expect learning methods to develop large number of rules. ELE modeling involves a large search space [Alcalá et al., 2011]. The MPG data concerns city-cycle fuel consumption in miles per gallon (mpg), to be predicted in terms of one multivalued discrete and five continuous attributes. The MPG6 dataset has 398 samples. The categorical data (ANA) is one of the data sets used in the book *Analyzing Categorical Data* by Jeffrey S. Simonoff. It contains information about the decisions taken by a supreme court. The ANA dataset concerns seven variables and 4052 samples. Abalone age data come from physical measurements. The abalone model has eight input variables and one output variable. The abalone dataset (ABA) contains 4177 samples. The ELE, MPG6, ANA and ABA datasets are available at `http://sci2s.ugr.es/keel/index.php`. The methods considered in [Alcalá et al., 2011] are summarized in Table 6.2. The method of Wang and Mendel (WM) is also a reference because PSA and the GFS use it as a rule generation procedure during evolution.

Table 6.2: Methods Considered by the Computational Experiments [Alcalá et al., 2011].

Method	**Type of learning**
WM(3)	rule base produced by WM, 3 linguistic labels for each variable
WM(5)	rule base produced by WM, 5 labels for each variable
WM(7)	rule base produced by WM, 7 labels for each variable
FSMOGFS	Gr. Lateral partition parameters, and rule base produced by WM
FSMOGFS+TUN	FSMOGFS + Tuning of MF parameters and rule selection by SPEA2
FSMOGFS^e+TUN^e	FSMOGFS+TUN including fast error estimation

6.2 Experiments and Results

The inputs parameters adopted by participatory search algorithms in the experiments reported in this section are population size of 60, and maximum number of function evaluations of 1000. Data sets were randomly divided into five folds, each partition containing 20% of the dataset. Four of these partitions are used for training and the remaining one is used for testing. The algorithms are run six times for each data partition using six distinct seeds.

The results show that the average mean-squared error for the test data achieved by the fuzzy models developed by PSAR, Table 6.3, is lower than the average mean-squared

error of test data achieved by the FSMOGFS^e+TUN^e, except for ANA data. Also, the *MSE* for the test data achieved by DPSA is lower than PSAR for ELE and MPG6 data. For the test data of ANA and ABA, FSMOGFS^e+TUN^e and PSAR achieve the lowest *MSE* value. Considering the test data PSAR, with WM using different number of labels for each linguistic variable, is more accurate than when the number of linguistic labels for each linguistic variable is kept fixed, WM(3),WM(5) and WM(7), respectively. Thus, PSAR performs better than FSMOGFS^e+TUN^e from the point of view of the test data of *MSE*.

Further analysis is pursued as suggested in [Derrac et al., 2011] and [Antonelli et al., 2013] to verify if there exist statistical differences among the performance of the algorithms. We use the same nonparametric tests as in Chapter 5.1. Recall that the confidence level is $\varepsilon = 0.05$. Table 6.4 shows how PSA and GFS are ranked. PSAR achieves the highest rank. Also, recall that the null hypothesis H_0 is that PSA and GFS algorithms are equivalent, that is, H_0 means that the rank of all algorithms are equal. If the hypothesis is rejected, then we conclude that the algorithms perform differently.

Iman-Davenport's test suggests that there are significant differences among the algorithms in all datasets once the null hypothesis is rejected (p-value$= 3.3E-4$). Thus the Holm post-hoc test is conducted with PSAR as the control algorithm. Table 6.5 shows that the Holm post-hoc test rejects the hypothesis concerning WM(3), WM(5), WM(7), PSST and DPST ($p < \varepsilon/i$), but do not reject FSMOGFS^e+TUN^e and DPSA. Therefore, PSAR outperforms WM(3), WM(5), WM(7), PSST and DPST because the rank pf PSAR is the highest and rejects the hypothesis in the Holm test. On the other hand, we notice that the difference of the performance of FSMOGFS^e+TUN^e, DPSA and PSAR are not statistical relevant because the null hypothesis is accepted.

Tabela 6.3: Average *MSE* of PSA and GFS Algorithms.

Algorithm	Dataset	ELE		MPG6		ANA		ABA	
	Measure	mean	sd	mean	sd	mean	sd	mean	sd
WM(3)	Rule	27		43		72		68	
	Tra	192241	9658	13.552	1.239	0.187	0.001	8.407	0.443
	Tst	192647	14436	14.649	3.204	0.189	0.005	8.422	0.545
WM(5)	Rule	65		115		124		199	
	Tra	56135	1498	4.136	0.317	0.027	0	3.341	0.13
	Tst	56359	4685	6.096	2.416	0.03	0.002	3.474	0.247
WM(7)	Rule	103		194		171		368	
	Tra	53092	1955	2.642	0.11	0.012	0	3.057	0.084
	Tst	55495	9452	6.382	2.126	0.017	0.003	3.268	0.185
FSMOGFS^e+TUN^e	Rule	8		20		10		8	
	Tra	9665	823	2.86	0.218	**0.003**	0	2.445	0.114
	Tst	10548	1150	4.562	0.714	**0.003**	0.001	2.509	0.184
PSAR	Rule	28		57		6		34	
	Tra	**9502**	3951.94	**1.6132**	1.3712	0.0256	0.0833	**1.67E-3**	2.95E-4
	Tst	10480	3986.8	1.5205	1.2983	0.0156	0.0574	**4.76E-3**	2.42E-4
PSST	Rule	27		77		4		23	
	Tra	11409	3874.74	3.424	1.3093	0.0292	0.0875	1.83E-3	1.82E-4
	Tst	10560	3906.8	3.1699	1.2639	0.0682	0.0485	5.01E-3	2.39E-4
DPSA	Rule	23		85		7		34	
	Tra	9586	3914.4058	1.8901	1.1779	0.0274	0.0913	1.83E-3	2.95E-4
	Tst	**10434**	3753.2432	**1.5151**	1.2563	0.0807	0.0866	4.76E-3	2.41E-4
DPST	Rule	23		63		5		30	
	Tra	11250	3020.37	2.0641	1.1532	0.0292	0.0836	5.5E-3	5.75E-4
	Tst	10544	3260.55	2.467	1.299	0.0795	0.0373	4.85E-3	2.74E-4

The analysis of Table 6.6 highlights that for each dataset, the average of execution

Table 6.4: Average rank of the algorithms.

Algorithm	Friedman rank	p-value	H_0
WM(3)	8		
WM(5)	6		
WM(7)	5.5		
FSMOGFS^e+TUN^e	3.75		
PSAR	1.875	3.3E-4	Rejected
PSST	4.75		
DPSA	2.125		
DPST	4		

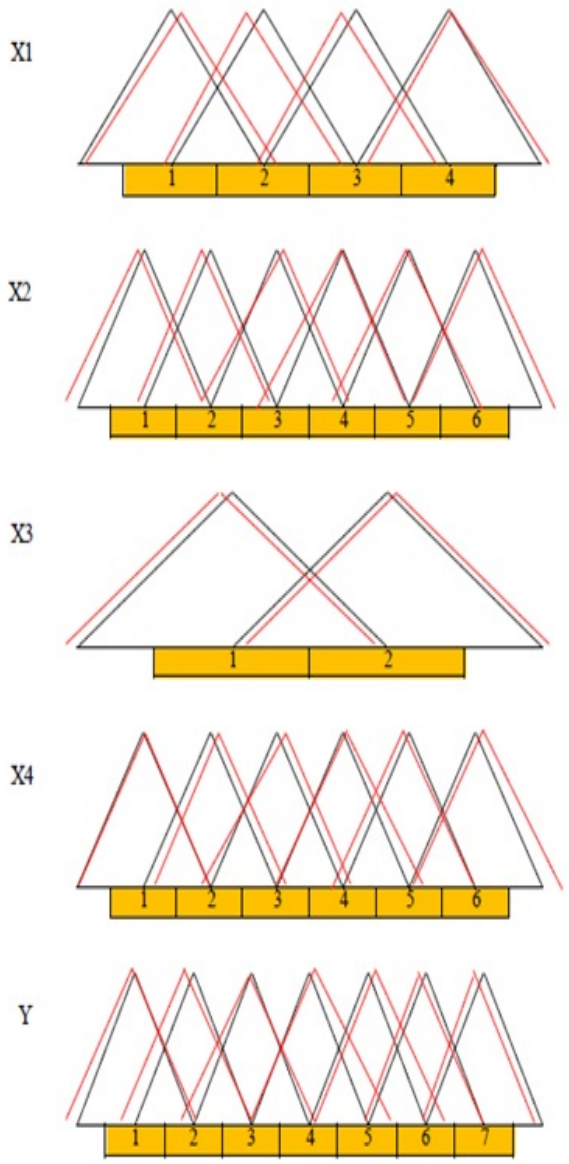
Table 6.5: Holm's Post-Hoc for $\varepsilon = 0.05$.

Control algorithm: PSAR					
i	Algorithm	z value	p-value	ε/i	H_0
7	WM(3)	3.5362	0.0004	0.0071	Rejected
6	WM(5)	2.3815	0.00723	0.0083	Rejected
5	WM(7)	2.0928	0.00363	0.01	Rejected
4	PSST	1.6598	0.0096	0.0125	Rejected
3	DPST	1.2268	0.02198	0.0166	Rejected
2	FSMOGFS^e+TUN^e	0.2790	0.2482	0.025	Accepted
1	DPSA	0.1443	0.8852	0.05	Accepted

of FSMOGFS^e+TUN^e and PSA in minutes and seconds. We can observe the different complexity of the solutions which generated during the evolutionary process. The computational cost of the fitness evaluation depends on the complexity of number of rules and number of conditions in the rules. In the case of ANA, the PSA needs less time than FSMOGFS^e+TUN^e due to the small number of rule. Further, it is more than 3 minutes in the case ANA and ABA due to the large number of samples.

Table 6.6: Average time of a run of the algorithms - (minutes and seconds - M:S)

Algorithm	ELE	MPG6	ANA	ABA
FSMOGFS^e+TUN^e	00:42	1:00	5:17	3:54
PSAR	00:45	00:53	5:05	4:25
PSST	00:42	1:02	5:06	3:57
DPSA	00:41	1:03	5:16	4:06
DPST	00:38	0:58	5:13	4:18



X1	X2	X3	X4	Y
1	1	1	1	1
1	2	1	1	1
1	1	1	2	2
1	2	1	2	2
1	2	2	2	3
2	1	1	2	3
2	2	2	3	3
2	4	2	2	3
2	4	3	3	5
2	4	2	1	3
2	3	2	2	4
2	3	1	1	1
2	1	1	2	2
3	3	2	3	4
3	2	1	3	3
3	4	3	2	4
3	5	3	4	6
3	6	4	5	7
3	3	2	4	4
3	2	2	2	3
3	3	2	2	4
3	6	4	3	6
3	6	4	2	6
3	6	3	2	4
3	2	1	2	3
4	2	1	2	3
4	4	2	2	4
4	2	2	2	4

Rule base: 28

MSE-tra: 9502

MSE-tst: 10480

Figure 6.4: Data and rule base developed by PSA from ELE data.

Figure 6.4 shows a data and rule base developed by PSA from ELE data. Examples of rules are:

rule 1: IF X1 is 1 and X2 is 1 and x3 is 1 and x4 is 1 THEN Y is 1

rule 2: IF X1 is 1 and X2 is 2 and x3 is 1 and x4 is 1 THEN Y is 1

⋮ ⋮ ⋮ ⋮

Summing up, the use of PSA to develop fuzzy rule-based models of actual electric maintenance, city-cycle fuel consumption in miles per gallon, analysis categorical, and age of abalone data illustrate its potential to solve complex problems. Overall, the results suggest that PSA with arithmetical-like recombination (PSAR) performs better than current state of the art genetic fuzzy system approaches from the point of view of the test

data of average mean square error. Participatory search algorithms are simpler, have high computational performance, and require few parameters to run. Thus, we may conclude that PSAR is a highly competitive population-based search approach.

6.3 Summary

Participatory search is a population-based instance of the participatory learning paradigm. The applications addressed in this chapter concerned the use of PSA to develop fuzzy linguistic models of actual data. The performance of the models produced by the PSA algoritms were evaluated and compared with a state of the art genetic fuzzy system approach. The results suggest that the participatory search algorithm with arithmetical-like recombination (PSAR) performs best.

Chapter 7

Conclusion

This work has introduced a new class of population-based search algorithms based on participatory learning. Participatory search is a population-based algorithm whose individuals affect the search process through individuals compatibility and an arousal mechanism. Compatibility and arousal are mechanisms to account for the influence of the individuals during search. Recombination arises from an instance of participatory learning formula, and mutation uses selected and recombined individuals to produce variations weighted by compatibility and arousal.

Analysis to the participatory search algorithm using random search theory suggests that the participatory search algorithm with arithmetical recombination converges to global optimum with probability one. It visits and keeps the global optimum in a finite number of steps, and the optimum is never lost regardless of initialization. Because all participatory search instances share the same foundations, a similar convergence behavior is expected from all the instances.

Results obtained from extensive computational experiments were developed to evaluate and compare the performance of the participatory search learning algorithms with population-based algorithms representative of the current state of the art in the area. In particular, the participatory search algorithm with arithmetical-like recombination performed the best. Nonparametric statistical tests also endorses that this instance of the participatory search performs best among the algorithms considered herein.

Applications of participatory search in fuzzy rule-based modeling using actual data illustrate the potential of participatory search to solve complex problems. Computational results suggest that the participatory search algorithm performs better than current state of the art genetic fuzzy system approach. In addition to its performance from the point of view of the quality of solutions they develop, participatory search algorithms are simpler, are faster, and require fewer parameters, which makes them very attractive in practice.

This work has introduced essential operational ant theoretical constructs for further development of participatory serch algorithms. The participatory search algorithm definition allows other tools to be added in the design stage. An advantage to design is that participatory search algorithm requires few parameters and it is very easy to merge with other metaheuristic algorithm to improve the performance. Further, the main aim in doing this is to balance between the exploration versus the exploitation behavior of these algorithms. Therefore, new opportunities emerge, which may lead to more powerful and flexible solution methods for optimization problems. In the near future, an important research topic is to combine multiple meta-heuristic algorithms to improve their individual performance.

References

[Agbali et al., 2010] Agbali, M., Reichard, M., Bryjová, A., Bryja, J., and Smith, C. (2010). Mate choice for nonadditive genetic benefits correlate with mhc dissimilarity in the rose bitterling (rhodeus ocellatus). *Evolution*, 64(6):1683–1696.

[Alcalá et al., 2007] Alcalá, R., Alcalá-Fdez, J., Herrera, F., and Otero, J. (2007). Genetic learning of accurate and compact fuzzy rule based systems based on the 2-tuples linguistic representation. *International Journal of Approximate Reasoning*, 44(1):45–64.

[Alcalá et al., 2011] Alcalá, R., Gacto, M. J., and Herrera, F. (2011). A fast and scalable multiobjective genetic fuzzy system for linguistic fuzzy modeling in high-dimensional regression problems. *IEEE Transactions on Fuzzy Systems*, 19(4):666–681.

[Alcalá-Fdez et al., 2009] Alcalá-Fdez, J., Sánchez, L., García, S., Ventura, S., del Jesus, M., Herrera, F., Otero, J., Garrell, J., Romero, C., Bacardit, J., Rivas, V., and Fernández, J. (2009). Keel: A software tool to assess evolutionary algorithms to data mining problems. *Soft Computing*, 13(3):307–318.

[Antonelli et al., 2013] Antonelli, M., Ducange, P., and Marcelloni, F. (2013). An efficient multi-objective evolutionary fuzzy system for regression problems. *International Journal of Approximate Reasoning*, 54(9):1434–1451.

[Baluja, 1994] Baluja, S. (1994). Population-based incremental learning. a method for integrating genetic search based function optimization and competitive learning. In *Technical report*, volume 94, Carnegie Mellon University Pittsburgh, USA.

[Birchenhall et al., 1997] Birchenhall, C., Kastrinos, N., and Metcalfe, S. (1997). Genetic algorithms in evolutionary modelling. *Journal of Evolutionary Economics*, 7(4):375–393.

[Birchenhall and shin Lin, 2000] Birchenhall, C. and shin Lin, J. (2000). Learning and adaptive artificial agents: Analysis of an evolutionary economic model. In *Computing in Economics and Finance*, number 327. University of Manchester, United Kingdom.

[Blum and Roli, 2003] Blum, C. and Roli, A. (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys*, 35(3):268–308.

[Brown, 2004] Brown, R. G. (2004). *Smoothing, forecasting and prediction of discrete time series*. Prentice-Hall, New Jersey, USA.

[Busch, 2005] Busch, J. W. (2005). The evolution of self-compatibility in geographically peripheral populations of leavenworthia alabamica (brassicaceae). *American Journal of Botany*, 92(9):1503–1512.

[Chen and Wang, 2010] Chen, Z.-Q. and Wang, R.-L. (2010). An efficient real-coded genetic algorithm for real-parameter optimization. *Sixth International Conference on Natural Computation*, 5:2276–2280.

[Chen and Yin, 2012] Chen, Z.-Q. and Yin, Y.-F. (2012). An new crossover operator for real-coded genetic algorithm with selective breeding based on difference between individuals. *Eighth International Conference on Natural Computation*, 5:2276–2280.

[Colorni et al., 1991] Colorni, A., Dorigo, M., Maniezzo, V., et al. (1991). Distributed optimization by ant colonies. In *Proceedings of the first European Conference on Artificial Life*, volume 142, pages 134–142, Paris, France.

[Cordón, 2001] Cordón, Ó. (2001). *Genetic fuzzy systems: Evolutionary tuning and learning of fuzzy knowledge bases*. Advances in Fuzzy Systems. World Scientific Publishing, Singapore.

[Cordón et al., 2001] Cordón, O., Herrera, F., Gomide, F., Hoffmann, F., and Magdalena, L. (2001). Ten years of genetic fuzzy systems: Current framework and new trends. In *IFSA World Congress and 20th NAFIPS International Conference*, volume 3, pages 1241–1246.

[Derrac et al., 2011] Derrac, J., García, S., Molina, D., and Herrera, F. (2011). A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation*, 1(1):3–18.

[Dorigo, 1992] Dorigo, M. (1992). Optimization, learning and natural algorithms. In *Ph. D. Thesis, System and Information Engineering*. Politecnico di Milano, Italy.

[Dorigo et al., 2006] Dorigo, M., Birattari, M., and Thomas, S. (2006). Ant colony optimization. *IEEE Computational Intelligence*, 1(4):28–39.

[Eberbach, 2005] Eberbach, E. (2005). Toward a theory of evolutionary computation. *BioSystems*, 82(1):1–19.

[Eberhart and Kennedy, 1995] Eberhart, R. C. and Kennedy, J. (1995). A new optimizer using particle swarm theory. In *Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, volume 1, pages 39–43, New York, USA.

[Eiben and Smith, 2015] Eiben, A. and Smith, J. (2015). *Introduction to evolutionary computing*. Springer-Verlag, Berlin, Germany.

[Eiben and Rudolph, 1999] Eiben, A. E. and Rudolph, G. (1999). Theory of evolutionary algorithms: A bird's eye view. *Theoretical Computer Science*, 229(1):3–9.

[Fazzolari et al., 2013] Fazzolari, M., Alcala, R., Nojima, Y., Ishibuchi, H., and Herrera, F. (2013). A review of the application of multiobjective evolutionary fuzzy systems: Current status and further directions. *IEEE Transactions on Fuzzy systems*, 21(1):45–65.

[Fogel, 1998] Fogel, D. (1998). *Evolutionary computation: The fossil record*. Wiley-IEEE Press, New York, USA.

[Fonseca et al., 2009] Fonseca, L., Barbosa, H., and Lemonge, A. (2009). A similarity-based surrogate model for enhanced performance in genetic algorithms. *Opsearch*, 46(1):89–107.

[Glover et al., 2000] Glover, F., Laguna, M., and Martí, R. (2000). Fundamentals of scatter search and path relinking. *Control and Cybernetics*, 29(3):653–684.

[Gupta, 1999] Gupta, M. M. (1999). *Soft computing and intelligent systems: Theory and applications*. Academic Press Series in Engineering, San Diego, USA.

[Herrera, 2008] Herrera, F. (2008). Genetic fuzzy systems: Taxonomy, current research trends and prospects. *Evolutionary Intelligence*, 1(1):27–46.

[Herrera and Martínez, 2000] Herrera, F. and Martínez, L. (2000). A 2-tuple fuzzy linguistic representation model for computing with words. *IEEE Transactions on Fuzzy Systems*, 8(6):746–752.

[Hwang, 1998] Hwang, H.-S. (1998). Control strategy for optimal compromise between trip time and energy consumption in a high-speed railway. *Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on Systems*, 28(6):791–802.

[Iman and Davenport, 1980] Iman, R. and Davenport, J. (1980). Approximations of the critical region of the friedman statistic. *Communications in Statistics-Theory and Methods*, 9(6):571–595.

[Ishibuchi et al., 2008] Ishibuchi, H., Narukawa, K., Tsukamoto, N., and Nojima, Y. (2008). An empirical study on similarity-based mating for evolutionary multiobjective combinatorial optimization. *European Journal of Operational Research*, 188(1):57–75.

[Ishibuchi and Shibata, 2003a] Ishibuchi, H. and Shibata, Y. (2003a). An empirical study on the effect of mating restriction on the search ability of emo algorithms. In *Evolutionary Multi-Criterion Optimization*, pages 433–447, Berlin, Germany. Springer-Verlag.

[Ishibuchi and Shibata, 2003b] Ishibuchi, H. and Shibata, Y. (2003b). *A similarity-based mating scheme for evolutionary multiobjective optimization*. Springer-Verlag, Berlin, Germany.

[Ishibuchi et al., 2007] Ishibuchi, H., Tsukamoto, N., and Nojima, Y. (2007). Choosing extreme parents for diversity improvement in evolutionary multiobjective optimization algorithms. In *Man and Cybernetics, 2007. ISIC. IEEE International Conference on Systems*, pages 1946–1951.

[Karnopp, 1963] Karnopp, D. (1963). Random search techniques for optimization problems. *Automatica*, 1(2):111–121.

[Kennedy et al., 1995] Kennedy, J., Eberhart, R., et al. (1995). Particle swarm optimization. In *Proceedings of IEEE International Conference on Neural Networks*, volume 4, pages 1942–1948, Perth, Australia.

[Kennedy et al., 2001] Kennedy, J. F., Kennedy, J., and Eberhart, R. C. (2001). *Swarm intelligence*. Morgan Kaufmann, San Francisco, USA.

[Khatib and Fleming, 1998] Khatib, W. and Fleming, P. J. (1998). *The stud GA: A mini revolution*. Springer-Verlag, Berlin, Germany.

[Klenke, 2013] Klenke, A. (2013). *Probability theory: A comprehensive course*, volume 10. Springer-Verlag, London, United Kingdom.

[Liang et al., 2013a] Liang, J., Qu, B., Suganthan, P., and Hernández-Díaz, A. G. (2013a). Problem definitions and evaluation criteria for the cec 2013 special session on real-parameter optimization. In *Technical Report 2012*, volume 2012, Singapore. Zhengzhou University and Nanyang Technological University.

[Liang et al., 2013b] Liang, J., Qu, B., Suganthan, P., and Hernández-Díaz, A. G. (2013b). Ranking results of cec'13 special session and competition on real-parameter single objective optimization. In *Technical Report 2012*, pages 1–11, Singapore. Zhengzhou University and Nanyang Technological University.

[Liu and Gomide, 2013a] Liu, Y. L. and Gomide, F. (2013a). Evolutionary participatory learning in fuzzy systems modeling. In *IEEE International Conference on Fuzzy Systems*, pages 1–8, Hyderabad, India.

[Liu and Gomide, 2013b] Liu, Y. L. and Gomide, F. (2013b). Fuzzy systems modeling with participatory evolution. In *IFSA World Congress and NAFIPS Annual Meeting (IFSA/NAFIPS)*, pages 380–385, Edmonton, AB, Canada.

[Liu and Gomide, 2013c] Liu, Y. L. and Gomide, F. (2013c). Participatory genetic learning in fuzzy system modeling. In *IEEE International Workshop on Genetic and Evolutionary Fuzzy Systems*, pages 1–7, Singapore.

[Liu and Gomide, 2016] Liu, Y. L. and Gomide, F. (2016). Participatory search algorithms in fuzzy modeling. In *Proc. World Conference in Soft Computing*, Berkeley, USA.

[Lomolino et al., 2006] Lomolino, M., Brett, R., and James, H. (2006). *Biogeography*. Sinauer Associates, Sunderland, USA.

[Meghanathan et al., 2012] Meghanathan, N., Chaki, N., and Nagamalai, D. (2012). *Advances in Computer Science and Information Technology. Computer Science and Engineering: Second International Conference*. Springer, Bangalore, India.

[Michalewicz, 1996] Michalewicz, Z. (1996). *Genetic algorithms+ data structures= evolution programs*. Springer, Charlotte, USA.

[Price et al., 2006] Price, K., Storn, R. M., and Lampinen, J. A. (2006). *Differential evolution: A practical approach to global optimization*. Springer-Verlag, Berlin, Germany.

[Radcliffe and Surry, 1994] Radcliffe, N. J. and Surry, P. D. (1994). Formal memetic algorithms. In *Evolutionary Computing: AISB Workshop*, pages 1–16. Springer-Verlag, Berlin, Germany.

[Rothlauf, 2011] Rothlauf, F. (2011). *Design of modern heuristics: Principles and application*. Springer-Verlag, Berlin, Germany.

[Royden and Fitzpatrick, 1988] Royden, H. and Fitzpatrick, P. (1988). *Real analysis*. Macmillan, New York, USA.

[Rudolph, 1994] Rudolph, G. (1994). Convergence analysis of canonical genetic algorithms. *IEEE Transactions on Neural Networks*, 5(1):96–101.

[Russell and Norvig, 2003] Russell, S. and Norvig, P. (2003). *Artificial intelligence a modern approach*. Prentice hall, New Jersey, USA.

[Sheskin, 2006] Sheskin, D. (2006). *Handbook of parametric and nonparametric statistical procedures*. Chapman Hall, New York, USA.

[Simon, 2008] Simon, D. (2008). Biogeography-based optimization. *IEEE Transactions on Evolutionary Computation*, 12(6):702–713.

[Simon, 2013] Simon, D. (2013). *Evolutionary optimization algorithms*. John Wiley & Sons, New Jersey, USA.

[Simon et al., 2011] Simon, D., Rarick, R., Ergezer, M., and Du, D. (2011). Analytical and numerical comparisons of biogeography-based optimization and genetic algorithms. *Information Sciences*, 181(7):1224–1248.

[Solis and Wets, 1981] Solis, F. and Wets, R. (1981). Minimization by random search techniques. *Mathematics of Operations Research*, 6(1):19–30.

[Sra et al., 2012] Sra, S., Nowozin, S., and Wright, S. (2012). *Optimization for machine learning*. Mit Press, London, United Kingdom.

[Storn and Price, 1997] Storn, R. and Price, K. (1997). Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11(4):341–359.

[Voget and Kolonko, 1998] Voget, S. and Kolonko, M. (1998). Multidimensional optimization with a fuzzy genetic algorithm. *Journal of Heuristics*, 4(3):221–244.

[Wang, 1994] Wang, L.-X. (1994). *Adaptive fuzzy systems and control: Design and stability analysis*. Prentice-Hall, Upper Saddle River, USA.

[Wang and Mendel, 1992] Wang, L.-X. and Mendel, J. M. (1992). Generating fuzzy rules by learning from examples. *Man and Cybernetics, IEEE Transactions on Systems*, 22(6):1414–1427.

[Whittaker and Fernández-Palacios, 2007] Whittaker, R. J. and Fernández-Palacios, J. M. (2007). *Island biogeography: Ecology, evolution, and conservation*. Oxford University Press, United Kingdom.

[Yager, 1990] Yager, R. (1990). A model of participatory learning. *Man and Cybernetics, IEEE Transactions on Systems*, 20(5):1229–1234.

[Yager, 2000] Yager, R. (2000). Participatory genetic algorithms. *BISC Group List*.

[Yao and Liu, 1996] Yao, X. and Liu, Y. (1996). Fast evolutionary programming. In *Proceedings of the Fifth Annual Conference on Evolutionary Programming*, pages 451–460. MIT Press, London, United Kingdom.

Appendix A

Test Problems for Population-Based Optimization

This appendix introduces the benchmark optimization problems to evaluate the proposed algorithm PSAR in the Chapter 4. The following definition and parameter of the problems are available in the literature [Simon, 2008].

A.1 Definition of the Test Functions

Test functions are minimization problems with global minimum $[0_1, ..., 0_D]$ and $D = 30$ dimensions. The search ranges are adopted in the literture [Simon, 2008] for all test functions.

1. Ackley Function

$$f_1(x) = -20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}) - \exp(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i))) + 20 + e \tag{A.1}$$

   The graphical presentation of this problem in two-dimensional space is given in Figure A.1A.1.1. Ackley function is non-separable, regular, multi-modal and the number of local minima increases exponentially with the problem dimension.

2. Griewank Function

$$f_2(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}}) + 1 \tag{A.2}$$

   The graphical presentation of this problem in two-dimensional space is given in Figure A.1A.1.2. Griewank function is non-separable, regular, multi-modal and the number of local minima increases exponentially with the problem dimension.

3. Quartic Function

$$f_3(x) = \sum_{i=1}^{n} i x_i^4 + random[0, 1) \tag{A.3}$$

The graphical presentation of this problem in two-dimensional space is given in Figure A.2A.2.1. Quartic function is separable, regular, unimodal and nosiy function, where $random[0, 1)$ is a uniformly distributed random variable in $[0, 1)$.

4. Rastrigin Function

$$f_4(x) = \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i) + 10] \tag{A.4}$$

The graphical presentation of this problem in two-dimensional space is given in Figure A.2A.2.2. Rastrigin function is separable, regular, multi-modal and the number of local minima increases exponentially with the problem dimension.

5. Rosenbrock Function

$$f_5(x) = \sum_{i=1}^{n} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2] \tag{A.5}$$

The graphical presentation of this problem in two-dimensional space is given in Figure A.3A.3.1. Rosenbrock function is unimodal, non-separble and regular function.

6. Schwefel 1.2 Function

$$f_6(x) = \sum_{i=1}^{n} (\sum_{j=1}^{i} x_j)^2 \tag{A.6}$$

The graphical presentation of this problem in two-dimensional space is given in Figure A.3A.3.2. Schwefel 1.2 function is unimodal, non-separble and regular function.

7. Schwefel 2.21 Function

$$f_7(x) = \max_i \{|x_i|, 1 \leq i \leq n\} \tag{A.7}$$

The graphical presentation of this problem in two-dimensional space is given in Figure A.4A.4.1. Schwefel 2.21 function is unimodal, non-separble and non-regular function.

8. Schwefel 2.22 Function

$$f_8(x) = \sum_{i=1}^{n} |x_i| + \prod_{i=1}^{n} |x_i| \tag{A.8}$$

The graphical presentation of this problem in two-dimensional space is given in Figure A.4A.4.2. Schwefel 2.22 function is non-separble, non-regular and multi-modal function.

9. Sphere Function

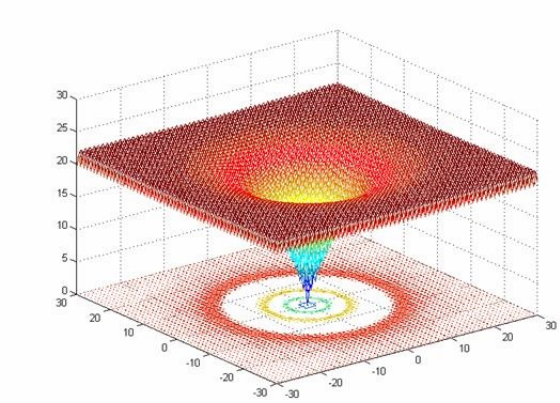
$$f_9(x) = \sum_{i=1}^{n} x_i^2 \tag{A.9}$$

The graphical presentation of this problem in two-dimensional space is given in Figure A.5A.5.1. Sphere function is separable, regular and unimodal function.

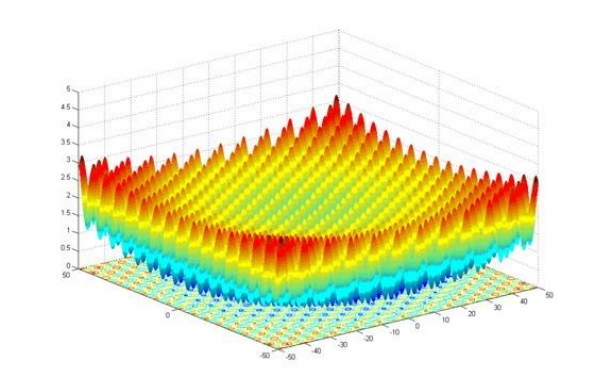
10. Step Function

$$f_{10}(x) = \sum_{i=1}^{n} \lfloor x_i + 0.5 \rfloor \tag{A.10}$$

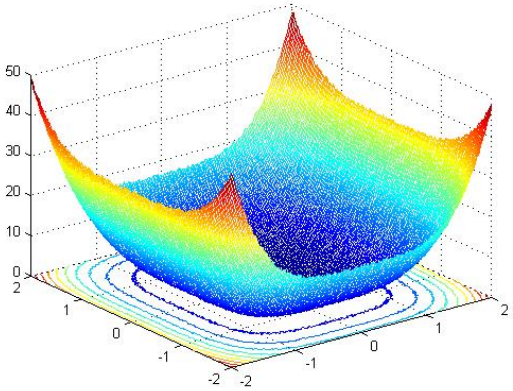
The graphical presentation of this problem in two-dimensional space is given in Figure A.5A.5.2. Step function is separable, non-regular, unimodal and discontinuous function.



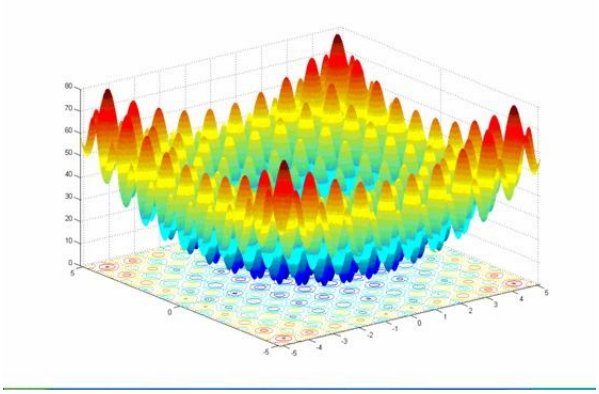
(A.1.1) Ackley Function.



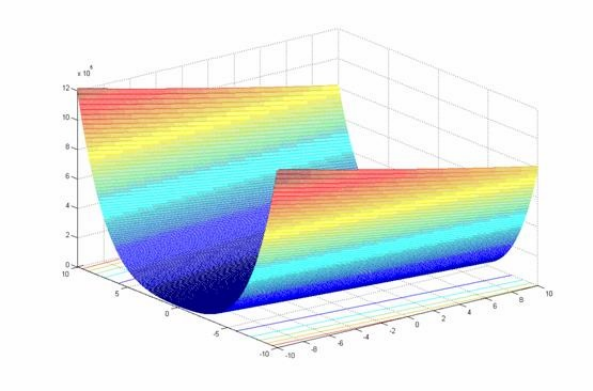
(A.1.2) Griewank Function.



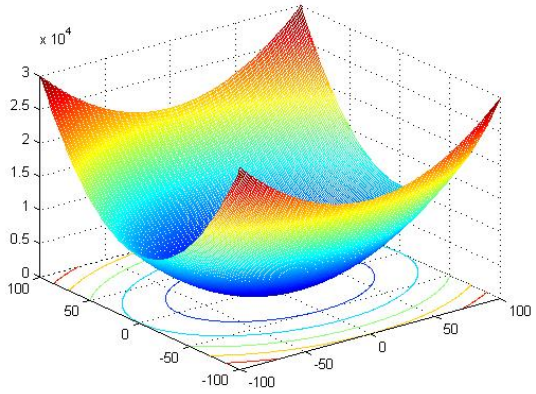
(A.2.1) Quartic Function.



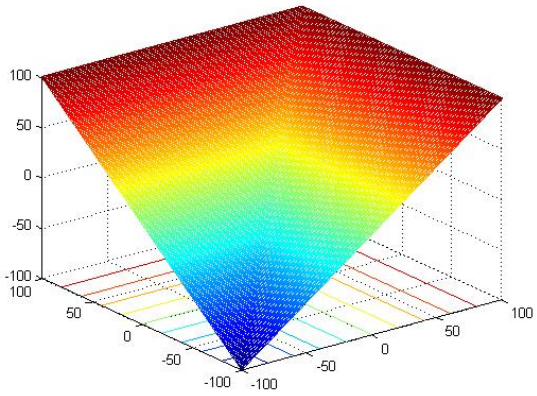
(A.2.2) Rastrigin Function.



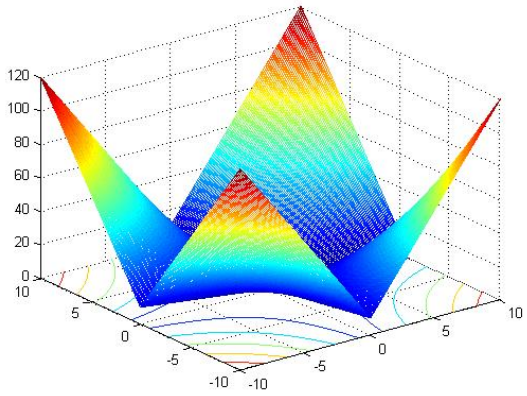
(A.3.1) Rosenbrock Function.



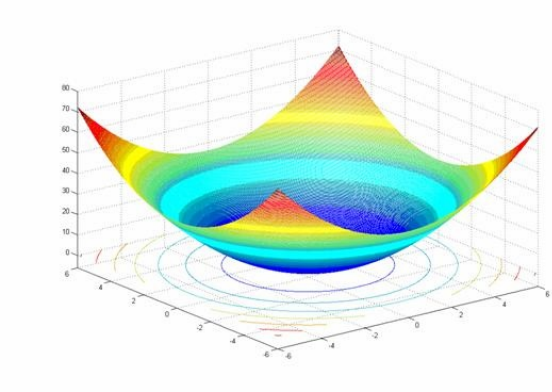
(A.3.2) Schwefel 1.2 Function.



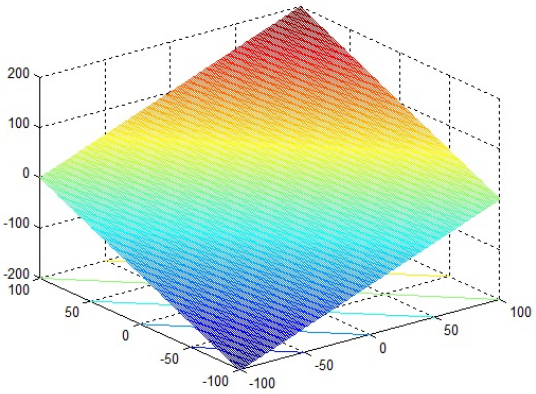
(A.4.1) Schwefel 2.21 Function.



(A.4.2) Schwefel 2.22 Function.



(A.5.1) Sphere Function.



(A.5.2) Step Function.

Appendix **B**

Test Problems for IEEE CEC 2013 competition

This appendix details the problems for the IEEE CEC 2013 on real-parameter optimization which included 28 benchmark functions to evaluate the proposed algorithm in the Chapter 4. The following definition and parameter of the problems are available in the technical report [Liang et al., 2013b].

B.1 Definition of the Test Functions

Test functions are minimization problems defined as following:

$$Minf(x), \quad x = [x_1, x_2, ..., x_D]^T$$

where D is dimension and $o = [o_1, o_2, ..., o_D]^T$ is the shifted global optimum, which is randomly distributed in $[-80, 80]^D$. The same search ranges are defined for all test functions $[-100, 100]^D$.

M is the rotation matrix generated from standard normally distributed by Gram-Schmidt orthonormalization. Λ^α is a diagonal matrix on D dimensions with the i^{th} diagonal elements as $\lambda_{ii} = \alpha^{\frac{i-1}{2(D-1)}}, \quad i = 1, 2, .., D.$

$T^\beta_{asy}: \quad$ if $\quad x_i > 0, \quad x_i = x_i^{1+\beta \frac{i-1}{D-1}\sqrt{x_i}}, \quad$ for $\quad i = 1, ..., D.$

$T_{osz}: \quad$ for $\quad x_i = sign(x_i) \exp(\widehat{x_i} + 0.049(\sin(c_i \widehat{x_i}) + \sin(c_2 \widehat{x_i}))), \quad$ for $\quad i = 1 \quad$ and $\quad D.$

where $\widehat{x_i} = \begin{cases} log(|x_i|) & \text{if} \quad x_i \neq 0 \\ 0 & \text{otherwise} \end{cases}$, $sign(x_i) = \begin{cases} -1 & \text{if} \quad x_i < 0 \\ 0 & \text{if} \quad x_i = 0 \\ 1 & \text{otherwise} \end{cases}$

$c_1 = \begin{cases} 10 & \text{if} \quad x_i > 0 \\ 5.5 & \text{otherwise} \end{cases}$, and $c_2 = \begin{cases} 7.9 & \text{if} \quad x_i > 0 \\ 3.1 & \text{otherwise} \end{cases}$

B.1.1 Unimodal Functions

1. f_1: Sphere Function

$$f_1(x) = \sum_{i=1}^{D} z_i^2 + f_1^* \quad , \quad z = x - o \tag{B.1}$$

The graphical presentation of this problem in two-dimensional space is given in Figure B.1B.1.1. Function f_1 is unimodal and separable function with $f_1^* = -1400$.

2. f_2: Rotated High Conditioned Elliptic Function

$$f_2(x) = \sum_{i=1}^{D} (10^6)^{\frac{i-1}{D-1}} z_i^2 + f_2^* \quad , \quad z = T_{osz}(M_1(x-o)) \tag{B.2}$$

The graphical presentation of this problem in two-dimensional space is given in Figure B.1B.1.2. Function f_2 is unimodal and non-separable function with $f_2^* = -1300$. The main feature of this function is quadratic ill-conditioned and smooth local irregularities.

3. f_3: Rotated Bent Cigar Function

$$f_3(x) = z_1^2 + 10^6 \sum_{i=2}^{D} z_i^2 + f_3^* \quad , \quad z = M_2 T_{asy}^{0.5}(M_1(x-o)) \tag{B.3}$$

The graphical presentation of this problem in two-dimensional space is given in Figure B.2B.2.1. Function f_3 is unimodal and non-separable function with $f_3^* = -1200$. The main feature of this function is smooth but narrow ridge.

4. f_4: Rotated Discus Function

$$f_4(x) = 10^6 z_1^2 + \sum_{i=2}^{D} z_i^2 + f_4^* \quad , \quad z = T_{osz}(M_1(x-o)) \tag{B.4}$$

The graphical presentation of this problem in two-dimensional space is given in Figure B.2B.2.2. Function f_4 is unimodal and non-separable function with $f_4^* = -1100$. The main feature of this function is asymmetrical and smooth local irregularities with one sensitive direction.

5. f_5: Different Powers Function

$$f_5(x) = \sqrt{\sum_{i=1}^{D} |z_i|^{2+4\frac{i-1}{D-1}}} + f_5^* \quad , \quad z = x - o \tag{B.5}$$

The graphical presentation of this problem in two-dimensional space is given in Figure B.3B.3.1. Function f_5 is unimodal and non-separable function with $f_5^* = -1000$. The main feature of this function is that sensitivities of the z_i-variables are different.

B.1.2 Basic Multimodal Functions

1. f_6: Rotated Rosenbrock's Function

$$f_6(x) = \sum_{i=1}^{D-1} (100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2) + f_6^* \quad , \quad z = M_1(\frac{2.048(x-o)}{100} + 100) \tag{B.6}$$

The graphical presentation of this problem in two-dimensional space is given in Figure B.3B.3.2. Function f_6 is unimodal and non-separable function with $f_6^* = -900$. The main feature of this function is multi-modal, non-separable and having a very narrow valley from local optimum to global optimum.

2. f_7: Rotated Schaffers Function

$$f_7(x) = (\frac{1}{D-1}\sum_{i=1} D - 1(\sqrt{z_i} + \sqrt{z_i}sin^2(50z_i^{0.2})))^2 + f_7^* \tag{B.7}$$

$$z_i = \sqrt{y_i^2 + y_{i+1}^2} \quad \text{for} \quad i = 1, ..., D, y = \Lambda^{10} M_2 T_{asy}^{0.5}(M_1(x-o))$$

The graphical presentation of this problem in two-dimensional space is given in Figure B.4B.4.1. Function f_7 is unimodal and non-separable function with $f_7^* = -800$. The main feature of this function is multi-modal, non-separable and asymmertical. Local optima's number is huge.

3. f_8: Rotated Ackley's Function

$$f_8(x) = -20\exp(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} z_i^2}) - \exp(\frac{1}{D}\sum_{i=1}^{D} cos(2\pi z_i)) + 20 + e + f_8^* \tag{B.8}$$

$$z = \Lambda^{10} M_2 T_{asy}^{0.5}(M_1(x-o))$$

The graphical presentation of this problem in two-dimensional space is given in Figure B.4B.4.2. Function f_8 is multi-modal, non-separable and asymmertical function with $f_8^* = -700$.

4. f_9: Rotated Weierstrass Function

$$f_9(x) = \sum_{i=1}^{D}(\sum_{k=0}^{k\max}[a^k \cos(2\pi b^k(z_i + 0.5))]) - D\sum_{k=0}^{k\max}[a^k \cos(2\pi b^k \cdot 0.5)] + f_9^* \tag{B.9}$$

$$a = 0.5, \quad b = 3, \quad k\max = 20, \quad z = \Lambda^{10} M_2 T_{asy}^{0.5}(M_1 \frac{0.5(x-o)}{100})$$

The graphical presentation of this problem in two-dimensional space is given in Figure B.5B.5.1. Function f_9 is multi-modal, non-separable and asymmetrical function with $f_9^* = -600$. The main feature of this function is continuous but differentiable only on a set of points.

5. f_{10}: Rotated Griewank's Function

$$f_{10}(x) = \sum_{i=1}^{D}\frac{z_i^2}{4000} - \prod_{i=1}^{D}\cos(\frac{z_i}{\sqrt{i}}) + 1 + f_{10}^* \tag{B.10}$$

$$z = \Lambda^{100} M_1 \frac{600(x-o)}{100}$$

The graphical presentation of this problem in two-dimensional space is given in Figure B.5B.5.2. Function f_{10} is multi-modal, non-separable and rotated function with $f_{10}^* = -500$.

6. f_{11}: Rastrigin's Function

$$f_{11}(x) = \sum_{i=1}^{D}(z_i^2 - 10\cos(2\pi z_i) + 10) + f_{11}^* \tag{B.11}$$

$$z = \Lambda^{10} T_{asy}^{0.2}(T_{osz}\frac{5.12(x-o)}{100})$$

The graphical presentation of this problem in two-dimensional space is given in Figure B.6B.6.1. Function f_{11} is multi-modal, separable and asymmertical function with $f_{11}^* = -400$. Local optima's number is huge.

7. f_{12}: Rotated Rastrigin's Function

$$f_{12}(x) = \sum_{i=1}^{D}(z_i^2 - 10\cos(2\pi z_i) + 10) + f_{12}^* \tag{B.12}$$

$$z = M_1\Lambda^{10}M_2(T_{osz}(M_1\frac{5.12(x-o)}{100}))$$

The graphical presentation of this problem in two-dimensional space is given in Figure B.6B.6.2. Function f_{12} is multi-modal, non-separable and asymmertical function with $f_{12}^* = -300$. Local optima's number is huge.

8. f_{13}: Non-continuous Rotated Rastrigin's Function

$$f_{13}(x) = \sum_{i=1}^{D}(z_i - 10\cos(2\pi z_i) + 10) + f_{13}^* \tag{B.13}$$

$$\widehat{x} = M_1\frac{5.12(x-o)}{100}, \quad z = M_1\Lambda^{10}M_2T_{asy}^{0.2}(T_{osz}(y))$$

$$y_i = \begin{cases} \widehat{x_i} & \text{if } |\widehat{x_i}| \leq 0.5 \text{ for } i = 1, 2, ..., D \\ round(2\widehat{x_i}) & \text{if } |\widehat{x_i}| > 0.5 \end{cases}$$

The graphical presentation of this problem in two-dimensional space is given in Figure B.7B.7.1. Function f_{13} is multi-modal, non-separable, Rotated and asymmertical function with $f_{13}^* = -200$. The main feature of this function is non-continuous and local optima's number is huge.

9. f_{14}: Schwefel's Function

$$f_{14}(z) = 418.9829 \times D - \sum_{i=1}^{D} g(z_i) + f_{14}^*$$

$$z = \Lambda^{10}(\frac{1000(x-o)}{100}) + 420.9687462275036$$

$$g(z_i) = \begin{cases} z_i \sin(|z_i|^{1/2}) & \text{if } |z_i| \leq 500 \\ (500 - mod(z_i, 500)\sin(\sqrt{|500 - mod(z_i, 500)|}) - \frac{(z_i-500)^2}{10000D} & \text{if } z_i > 500 \\ (mod(|z_i, 500) - 500)\sin(\sqrt{|mod(|z_i|, 500) - 500|}) - \frac{(z_i+500)^2}{10000D} & \text{if } z_i < -500 \end{cases} \tag{B.14}$$

The graphical presentation of this problem in two-dimensional space is given in Figure B.7B.7.2. Function f_{14} is multi-modal, non-separable, Rotated and asymmertical function with $f_{14}^* = -100$. The main feature of this function is that local optima's number is huge and second better local optimum is far from the global optimum.

10. f_{15}: Rotated Schwefel's Function

$$f_{15}(z) = 418.9829 \times D - \sum_{i=1}^{D} g(z_i) + f_{15}^*$$

$$z = \Lambda^{10} M_1 (\frac{1000(x-o)}{100}) + 420.9687462275036$$

$$g(z_i) = \begin{cases} z_i \sin(|z_i|^{1/2}) & \text{if } |z_i| \leq 500 \\ (500 - mod(z_i, 500) \sin(\sqrt{|500 - mod(z_i, 500)|}) - \frac{(z_i - 500)^2}{10000D} & \text{if } z_i > 500 \\ (mod(|z_i, 500) - 500) \sin(\sqrt{|mod(|z_i|, 500) - 500|}) - \frac{(z_i + 500)^2}{10000D} & \text{if } z_i < -500 \end{cases} \quad \text{(B.15)}$$

The graphical presentation of this problem in two-dimensional space is given in Figure B.8B.8.1. Function f_{15} is multi-modal, non-separable and asymmertical function with $f_{15}^* = 100$. The main feature of this function is that local optima's number is huge and second better local optimum is far from the global optimum.

11. f_{16}: Rotated Katsuura Function

$$f_{16}(x) = \frac{10}{D^2} \prod (1 + i \sum_{j=1}^{32} \frac{|2^j z_i - round(2^j z_i)|}{2^j})^{\frac{10}{D^{1.2}}} - \frac{10}{D^2} + f_{16}^* \quad \text{(B.16)}$$

$$z = M_2 \Lambda^{10} (M_1 \frac{5(x-o)}{100})$$

The graphical presentation of this problem in two-dimensional space is given in Figure B.8B.8.2. Function f_{16} is multi-modal, non-separable and asymmertical function with $f_{16}^* = 200$. The main feature of this function is continuous everywhere yet differentiable nowhere.

12. f_{17}: Lunacek bi-Rastrigin Function

$$f_{17}(x) = \min(\sum_{i=1}^{D} (\widehat{x}_i - \mu_0)^2, dD + s \sum_{i=1}^{D} (\widehat{x}_i - \mu_1)^2) + 10(D - \sum_{i=1}^{D} \cos(2\pi \widehat{z}_i)) + f_{17}^* \quad \text{(B.17)}$$

$$\mu_0 = 2.5, \quad \mu_1 = -\sqrt{\frac{\mu_0^2 - d}{s}}, \quad s = 1 - \frac{1}{2\sqrt{D+20} - 8.2}, \quad d = 1$$

$$y = \frac{10(x-o)}{100}, \quad \widehat{x}_i = 2 sign(x_i^*) y_i + \mu_0, \quad \text{for } i = 1, 2, ..., D$$

$$z = \Lambda^{100} (\widehat{x} - \mu_0)$$

The graphical presentation of this problem in two-dimensional space is given in Figure B.9B.9.1. Function f_{17} is multi-modal and asymmertical function with $f_{17}^* = 300$.

13. f_{18}: Rotated Lunacek bi-Rastrigin Function

$$f_{18}(x) = \min(\sum_{i=1}^{D}(\widehat{x}_i - \mu_0)^2, dD + s\sum_{i=1}^{D}(\widehat{x}_i - \mu_1)^2) + 10(D - \sum_{i=1}^{D}\cos(2\pi\widehat{z}_i)) + f_{17}^* \quad \text{(B.18)}$$

$$\mu_0 = 2.5, \quad \mu_1 = -\sqrt{\frac{\mu_0^2 - d}{s}}, \quad s = 1 - \frac{1}{2\sqrt{D+20} - 8.2}, \quad d = 1$$

$$y = \frac{10(x-o)}{100}, \quad \widehat{x}_i = 2sign(x_i^*)y_i + \mu_0, \quad \text{for } i = 1, 2, ..., D$$

$$z = M_2\Lambda^{100}(M_1\frac{\widehat{x} - \mu_0}{})$$

The graphical presentation of this problem in two-dimensional space is given in Figure B.9B.9.2. Function f_{18} is multi-modal, non-separable and asymmertical function with $f_{18}^* = 400$. The main feature of this function is continuous everywhere yet differentiable nowhere.

14. f_{19}: Rotated Expanded Griewank's plus Rosenbrock Function

$$\begin{array}{ll} \text{Basic Griewank's Function:} & g_1(x) = \sum_{i=1}^{D}\frac{x_i^2}{4000} - \prod_{i=1}^{D}\cos(\frac{x_i}{\sqrt{i}}) + 1 \\ \text{Basic Rosenbrock's Function:} & g_2(x) = \sum_{i=1}^{D-1}(100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2) \end{array}$$

$$f_{19}(x) = g_1(g_2(z_1, z_2)) + g_1(g_2(z_2, z_3)) + ... + g_1(g_2(z_{D-1}, z_D)) + g_1(g_2(z_D, z_1)) + f_{19}^* \quad \text{(B.19)}$$

$$z = M_1(\frac{5(x-o)}{100}) + 1$$

The graphical presentation of this problem in two-dimensional space is given in Figure B.10B.10.1. Function f_{19} is multi-modal and non-separable function with $f_{19}^* = 500$.

15. f_{20}: Rotated Expanded Scaffer's Function

$$\text{Scafffer's Function:} g(x, y) = 0.5 + \frac{(sin^2\sqrt{x^2 + y^2} - 0.5)}{(1 + 0.001(x^2 + y^2))^2)}$$

$$f_{20}(x) = g(z_1, z_2) + g(z_2, z_3) + ... + g(z_{D-1}, z_D) + g(z_D, z_1) + f_{20}^* \quad \text{(B.20)}$$

$$z = M_2T_{asy}^{0.5}(M_1(x - o))$$

The graphical presentation of this problem in two-dimensional space is given in Figure B.10B.10.2. Function f_{20} is multi-modal, non-separable and asymmetrical function with $f_{20}^* = 600$.

B.1.3 Composition Functions

$$f(x) = \sum_{i=1}^{n} \{\omega_i^* [\lambda_i g_i(x) + bias_i]\} + f^* \tag{B.21}$$

$f(x)$**:** new composition function

$g_i(x)$**:** i^{th} basic function used to construct the composition function

n**:** number of basic functions

o_i**:** new shifted optimum position for each $g_i(x)$, define the global and local optima's position

bias$_i$**:** define which optimum is global optimum

σ_i**:** used to control each $g_i(x)$'s coverage range, a small σ_i give a narrow range for the $g_i(x)$

λ_i**:** used to control each $g_i(x)$'s height

w_i**:** weight value for each $g_i(x)$, calculated as below:

$$w_i = \frac{1}{\sqrt{\sum_{j=1}^{D}(x_j - o_{ij})^2}} exp(-\frac{\sum_{j=1}^{D}(x_j - o_{ij})^2}{2D\sigma_i^2}) \tag{B.22}$$

Then normalize the weight $\omega_i = w_i / \sum_{i=1}^{n} w_i$

So when $x = o_i$, $\omega_j = \begin{cases} 1 & j = i \\ 0 & j \neq i \end{cases}$ for $j = 1, 2, ..., n, f(x) = bias_i + f^*$.
The local optimum which has the smallest bias values is global optimum. Functions $fi' = fi - f_i^*$ are used as g_i. In this way, the function values of blobal optima of g_i are equal to 0 for all composition functions in this work.

1. f_{21}: Composition Function 1
   $n = 5, \quad \sigma = [10, 20, 30, 40, 50]$
   $\lambda = [1, 1e-6, 1e-26, 1e-6, 0.1]$
   $bias = [0, 100, 200, 300, 400]$
   g_1: Rotated Rosenbrock's Function f_6'
   g_2: Rotated Different Powers Function f_5'
   g_3: Rotated Bent Cigar Function f_3'
   g_4: Rotated Discus Function f_4'
   g_5: Sphere Function f_1'
   The graphical presentation of this problem in two-dimensional space is given in Figure B.11B.11.1. Function f_{21} is multi-modal, non-separable and asymmetrical function with $f_{21}^* = 700$. The main feature of this function is different properties around different local optima.

2. f_{22}: Composition Function 2
   $n = 3$
   $\sigma = [20, 20, 20]$
   $\lambda = [1, 1, 1]$
   $bias = [0, 100, 200]$
   g_{1-3}: Schwefel's Function f'_{14}
   The graphical presentation of this problem in two-dimensional space is given in Figure B.11B.11.2. Function f_{22} is multi-modal, separable and asymmetrical function with $f^*_{22} = 800$. The main feature of this function is different properties around different local optima.

3. f_{23}: Composition Function 3
   $n = 3$
   $\sigma = [20, 20, 20]$
   $\lambda = [1, 1, 1]$
   $bias = [0, 100, 200]$
   g_{1-3}: Rotated Schwefel's Function f'_{15}
   The graphical presentation of this problem in two-dimensional space is given in Figure B.12B.12.1. Function f_{23} is multi-modal, non-separable and asymmetrical function with $f^*_{23} = 900$. The main feature of this function is different properties around different local optima.

4. f_{24}: Composition Function 4
   $n = 3$
   $\sigma = [20, 20, 20]$
   $\lambda = [0.25, 1, 2.5]$
   $bias = [0, 100, 200]$
   g_1: Rotated Schwefel's Function f'_{15}
   g_2: Rotated Rastrigin's Function f'_{12}
   g_3: Rotated Weierstrass Function f'_9
   The graphical presentation of this problem in two-dimensional space is given in Figure B.12B.12.2. Function f_{24} is multi-modal, non-separable and asymmetrical function with $f^*_{24} = 1000$. The main feature of this function is different properties around different local optima.

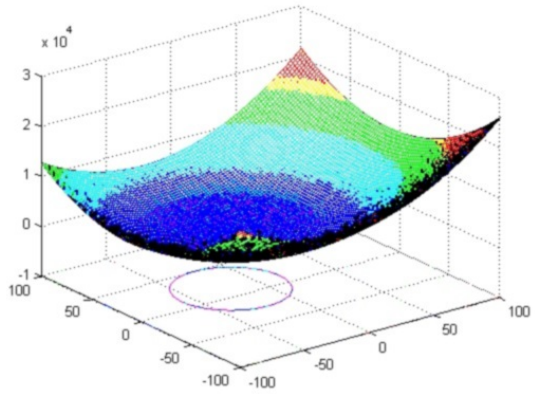
5. f_{25}: Composition Function 5
   All settings are same as Composition Function 4, except $\sigma = [10, 30, 50]$
   The graphical presentation of this problem in two-dimensional space is given in Figure B.13B.13.1. Function f_{25} is multi-modal, non-separable and asymmetrical function with $f^*_{25} = 1100$. The main feature of this function is different properties around different local optima.

6. f_{26}: Composition Function 6
   $n = 5$ $\sigma = [10, 10, 10, 10, 10]$
   $\lambda = [0.25, 1, 1e-7, 2.5, 10]$
   $bias = [0, 100, 200, 300, 400]$
   g_1: Rotated Schwefel's Function f'_{15}
   g_2: Rotated Rastrigin's Function f'_{12}
   g_3: Rotated High Conditioned Elliptic Function f'_2
   g_4: Rotated Weierstrass Function f'_9

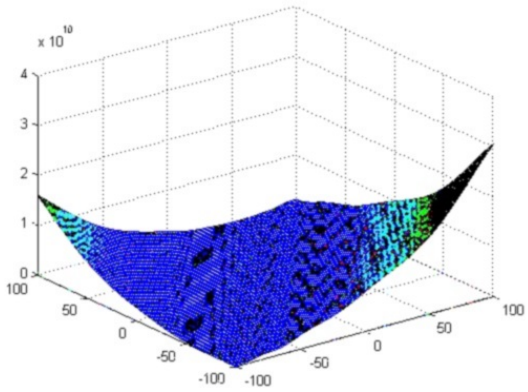
g_5: Rotated Griewank's Function f'_{10}
The graphical presentation of this problem in two-dimensional space is given in Figure B.13B.13.2. Function f_{26} is multi-modal, non-separable and asymmetrical function with $f^*_{26} = 1200$. The main feature of this function is different properties around different local optima.

7. f_{27}: Composition Function 7
   $n = 5$ $\sigma = [10, 10, 10, 20, 20]$
   $\lambda = [100, 10, 2.5, 25, 0.1]$
   $bias = [0, 100, 200, 300, 400]$
   g_1: Rotated Griewank's Function f'_{10}
   g_2: Rotated Rastrigin's Function f'_{12}
   g_3: Rotated Schwefel's Function f'_{15}
   g_4: Rotated Weierstrass Function f'_9
   g_5: Sphere Function f'_1
   The graphical presentation of this problem in two-dimensional space is given in Figure B.14B.14.1. Function f_{27} is multi-modal, non-separable and asymmetrical function with $f^*_{27} = 1300$. The main feature of this function is different properties around different local optima.

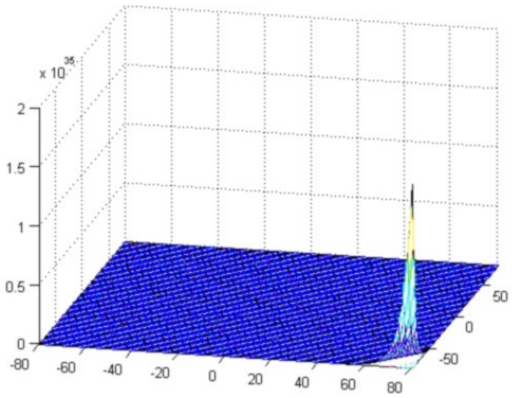
8. f_{28}: Composition Function 8
   $n = 5$ $\sigma = [10, 20, 30, 40, 50]$
   $\lambda = [2.5, 2.5e-3, 2.5, 5e-4, 0.1]$
   $bias = [0, 100, 200, 300, 400]$
   g_1: Rotated Expanded Griewank's plus Rosenbrock's Function f'_{19}
   g_2: Rotated Schaffer's Function f'_7
   g_3: Rotated Schwefel's Function f'_{15}
   g_4: Rotated Expanded Scaffer's Function f'_{20}
   g_5: Sphere Function f'_1
   The graphical presentation of this problem in two-dimensional space is given in Figure B.14B.14.2. Function f_{28} is multi-modal, non-separable and asymmetrical function with $f^*_{28} = 1400$. The main feature of this function is different properties around different local optima.



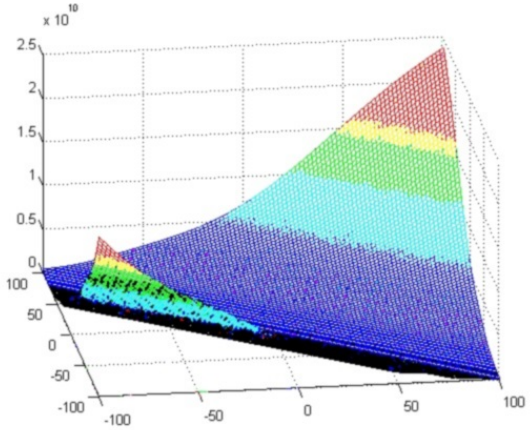
(B.1.1) f1



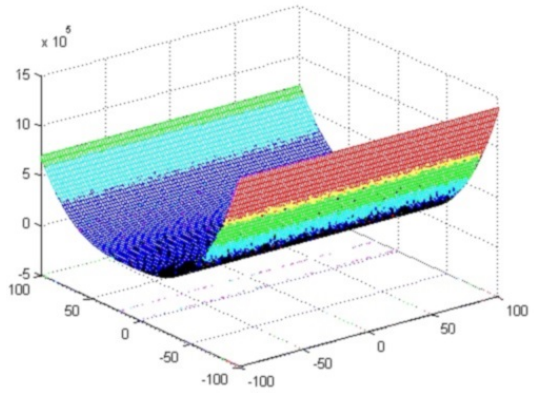
(B.1.2) f2



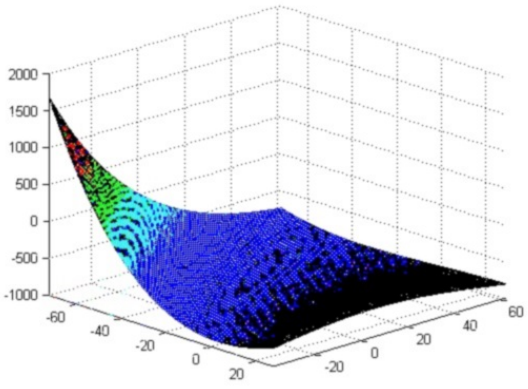
(B.2.1) f3



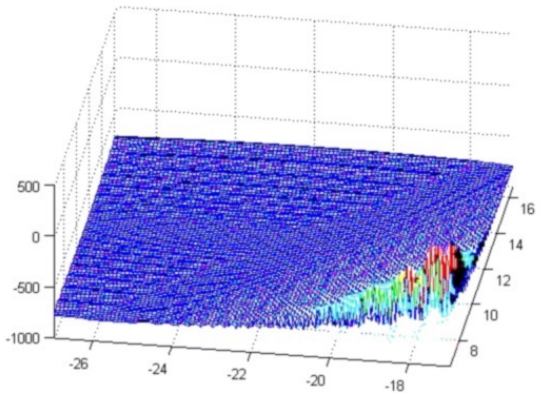
(B.2.2) f4



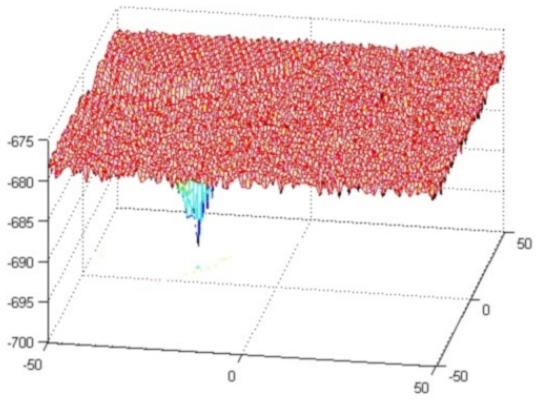
(B.3.1) f5



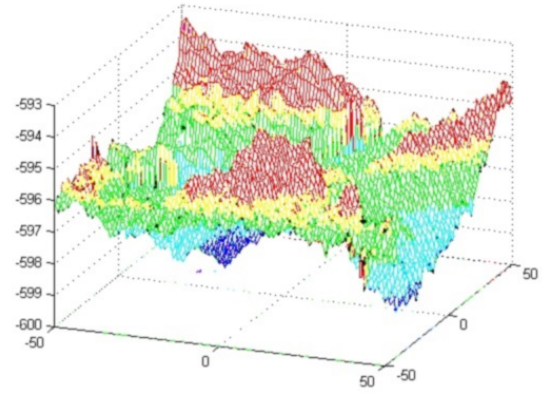
(B.3.2) f6



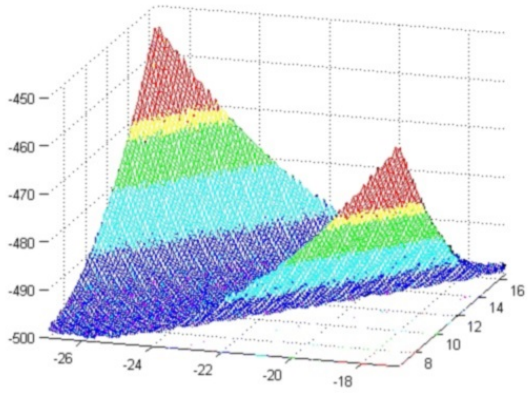
(B.4.1) f7



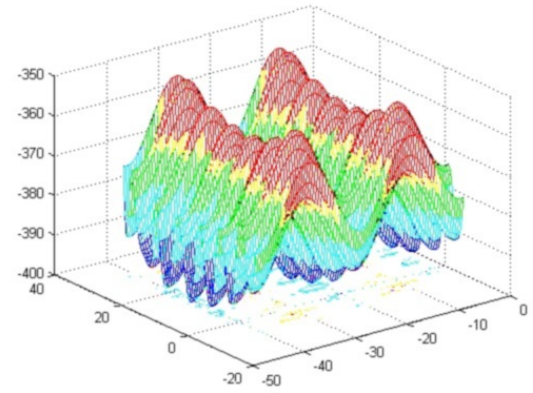
(B.4.2) f8



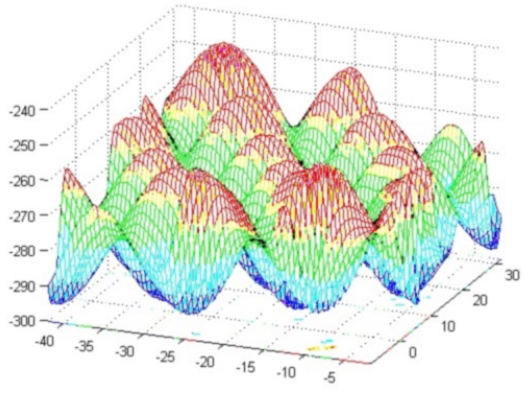
(B.5.1) f9



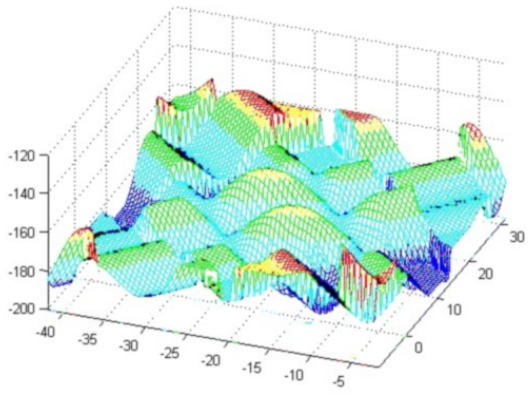
(B.5.2) f10



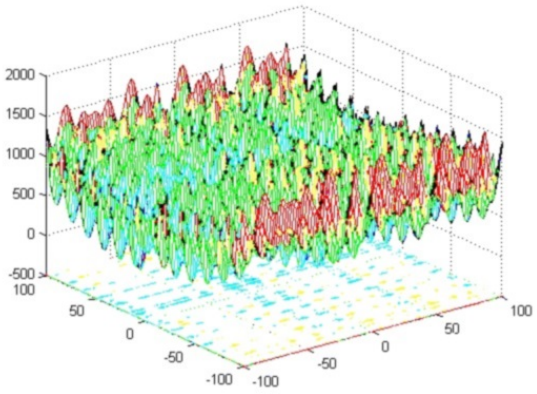
(B.6.1) f11



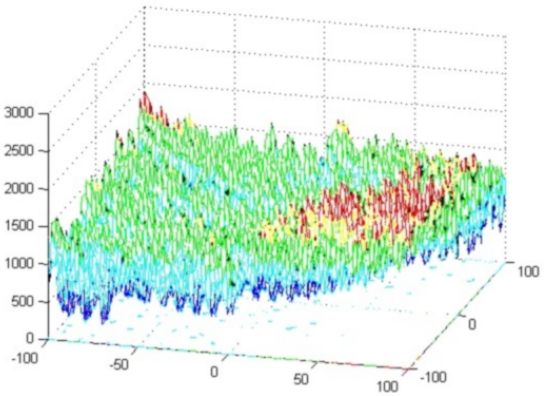
(B.6.2) f12



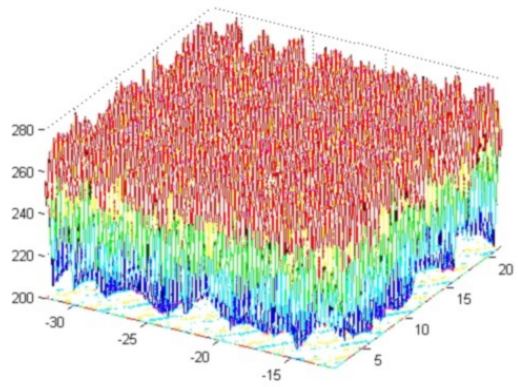
(B.7.1) f13



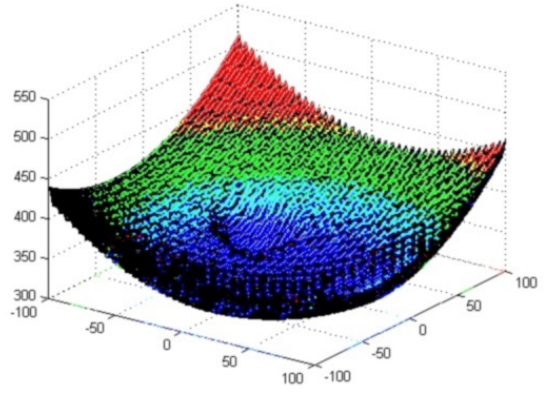
(B.7.2) f14



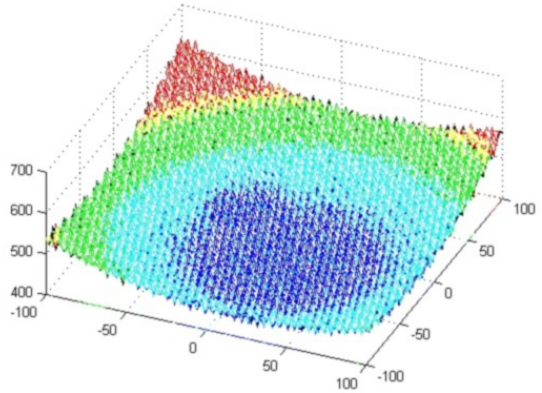
(B.8.1) f15



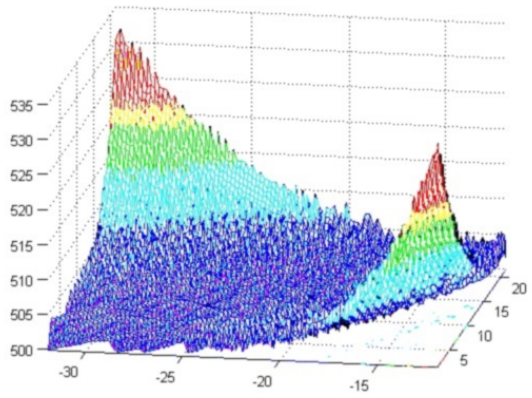
(B.8.2) f16



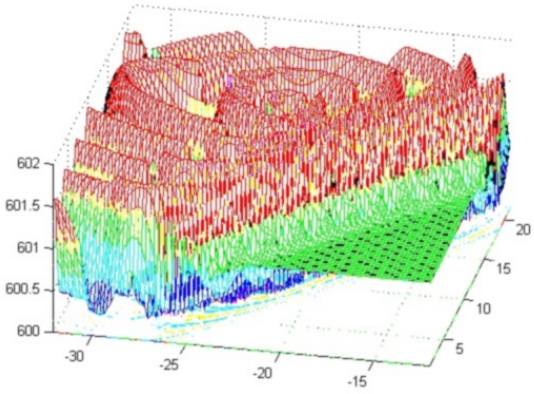
(B.9.1) f17



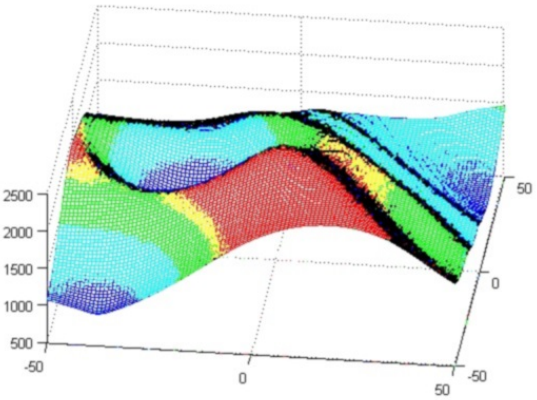
(B.9.2) f18



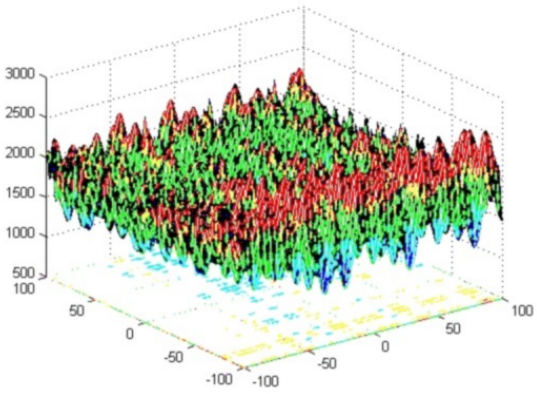
(B.10.1) f19



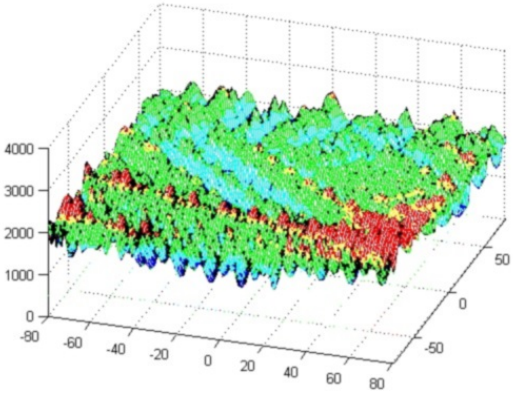
(B.10.2) f20



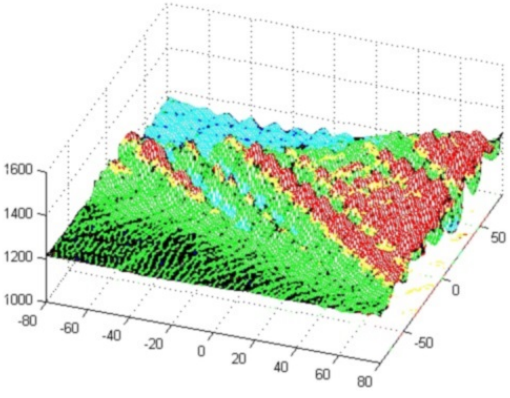
(B.11.1) f21



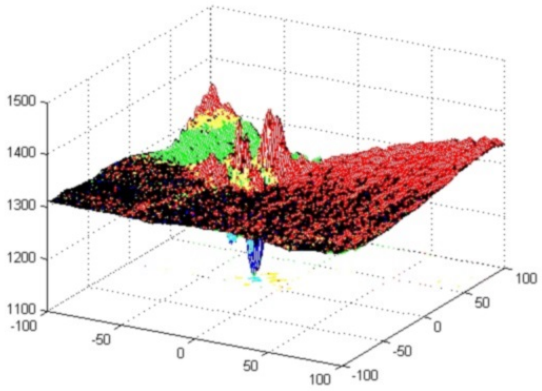
(B.11.2) f22



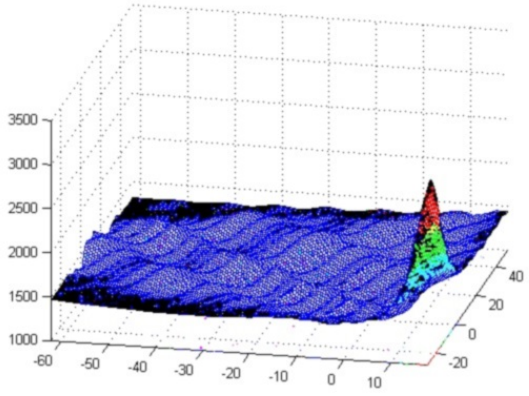
(B.12.1) f23



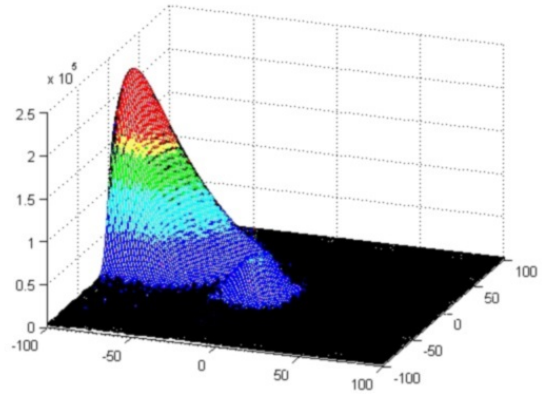
(B.12.2) f24



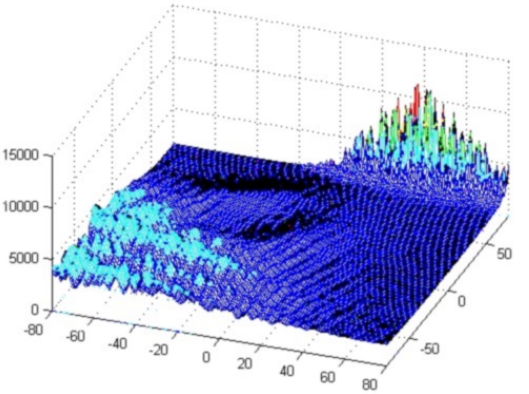
(B.13.1) f25



(B.13.2) f26



(B.14.1) f27



(B.14.2) f28

Figure B.14: Test functions on real-parameter optimization for IEEE CEC 2013.