PARTICLE SWARM SYSTEMS FOR MULTIMODAL OPTIMIZATION

by
Murat Yılmaz

Submitted to the Institute of Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science
in
Computer Engineering

Yeditepe University
2006

PARTICLE SWARM SYSTEMS FOR MULTIMODAL OPTIMIZATION

APPROVED BY:

Assist.Prof.Dr. Ender Özcan ……
(Thesis Supervisor)

Assoc.Prof.Dr. M.Kudret Yurtseven ……

Assoc.Prof.Dr. Raşit Eskicioğlu ……

DATE OF APPROVAL: 02/08/2006

ACKNOWLEDGEMENTS

I would like to thank to my thesis supervisor Assist.Prof. Dr. Ender Özcan. His motivation and willpower kept me doing all the work.

Thanks to my wife, Burcu Göksedef Yılmaz, for her being.

ABSTRACT

Many real world problems in science, engineering and economy, involve in optimization. Multimodal optimization problems represent a subset of such problems, where the goal is to obtain multiple different solutions with equal (or preferable) quality in a single search space. Meta-heuristics, especially, evolutionary computation techniques are the most commonly used approaches for solving difficult optimization problems. However, due to the multimodality, they might require some enhancements. Particle Swarm Optimizer (PSO), as a swarm intelligence technique, has proven its success in solving optimization problems. Additionally, several modified versions of the PSO algorithm make it a good choice for attacking multimodal problems as well. In this thesis, the performance of existing multimodal PSO techniques are analyzed over a set of well-known benchmark functions. Moreover, a new PSO algorithm based on craziness and hill-climbing is proposed for solving multimodal optimization problems.

ÖZET

Bilim, mühendislik ve ekonomi gibi birçok alandaki yaşamsal problem optimizasyon ile iç içedir. Bu optimizasyon problemlerinin bir kısmı, amacın bir arama uzayında birbirinden farklı ve eşit kalitedeki birden fazla çözüm kümelerini bulmak olan çok doruklu problemlerden oluşur. Zor optimizasyon problemlerini çözmek için genel olarak meta-buluşsal yöntem ve yaklaşımlar kullanılır. Ancak, problemin çok doruklu olduğu durumlarda bazı iyileştirmeler gerekebilir. Bir Swarm Intelligence tekniği olan Particle Swarm Optimizer (PSO) yönteminin en uygun çözüm kümesini bulmakta başarılı olduğu deneysel olarak gösterilmiştir. Ayrıca, PSO'nun çeşitli değişikliklere uğratılmış sürümleri de çok doruklu problemlerde kullanılabilirliğini sağlamıştır. Bu tez, çok doruklu problemler için önerilmiş mevcut PSO algoritmalarının yaygın kullanılan karşılaştırma fonksiyonları üzerindeki performanslarını karşılaştırmaktadır. Buna ek olarak, çok doruklu problemler için delilik içeren ve tepe tımanmadan faydalanan yeni bir PSO algoritması önerilmektedir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

Many real-world problems involve optimization of several parameters that interact in highly non-linear ways. Optimization techniques play an important role in science and technology to produce a more desirable outcome. Many recent advances in science, economics and engineering depend on optimization techniques for the determination of the optimal solutions.

Particle Swarm Optimization (PSO) is a new technique that seeks optimal solution(s) by simulating swarm behavior. Numerous researches have shown that PSO is a very effective optimization algorithm that outperforms various other algorithms on unimodal problems in global optimization [1][2][3]. However, there are a set of problems requiring the discovery of equal quality solutions, so that a user could make a choice in between them. In some problems, local optima, or a set of solutions with a predetermined quality levels, can also be requested. Multimodal optimization problems represent such class of problems in which researchers are interested. Over the years, many algorithms have been introduced and new approaches have been implemented resulting in powerful optimization algorithms. As the complexity of the problems increases, more powerful techniques are needed to be generated not only for unimodal, but also for multimodal functions.

Different PSO algorithms have been already proposed for solving multimodal problems. In this thesis, a new PSO algorithm with craziness and hill-climbing for multimodal optimization, denoted as CPSO is presented and compared to the existing multimodal PSO algorithms; Species-based PSO, Niching PSO, nbest PSO, Stretching PSO and Unified PSO. Furthermore, the performance of the Niching PSO is improved with a simple modification. A multimodal problem can be a maximization or a minimization problem. In this thesis, an optimum will be refered as a *peak* regardless of the problem type.

In the next chapter, an introduction to the theory of optimization, evolutionary computation and swarm intelligence is provided. Additionally, PSO algorithm is described and a modification to PSO that guarantees convergence is presented. In the third chapter,

multimodal optimization is discussed. The existing and proposed PSO algorithms for multimodal optimization are explained in detail. In the fourth chapter, experimental settings and the results of the experiments are given. Finally, in the fifth chapter, a summary of the findings of this thesis and some future research topics are examined.

2. OPTIMIZATION

Optimization is part of our lives. Many scientific, engineering and economic problems have parameters that should be adjusted to produce more advantageous results. Optimization is the determination of the values of these parameters to provide optimality. The task of optimization has a very important role on many professions, such as, engineering and economy. Economists, engineers and operation researchers have to think about the optimal allocation of resources. These problems may be linear, where linear optimization algorithms are used, or may be non-linear. In this thesis, particle swarm algorithm, which is used for the optimization of non-linear problems, is examined.

2.1. Function Optimization

Function optimization deals with finding an optimal solution or solutions to a problem which is defined by a cost or an objective function. The optimization is done by systematically choosing the values of real or integer variables from within an allowed set. Formally, for an objective function $f(x)$ over an n-dimensional space S, where $x \in S$; the purpose of the optimization is to find a solution x^* such that, $f(x^*) \leq f(x)$ for all $x \in S$ for minimization problems, or $f(x^*) \geq f(x)$ for all $x \in$ S for maximization problems. The space S is also called as the search space, and the elements of S are called the candidate solutions.

Usually, the solutions that have the optimum value over the whole search space, the global optima x^*, as defined in previous paragraph, are of interest. When the global optima can not be found, then it may be acceptable to find a local optima. The local optimum x_l^* is defined as, $f(x_l^*) \leq f(x)$ for all $x \in L$ and $L \subset S$.

If the values of x is subject to hold some given equality or inequality constraints, the unconstrained optimization problem becomes a constrained optimization problem. That is, the optimization algorithm should take into consideration the limitations that have been specified by the constraints, while searching the search space.

If an objective function has only one optimum point and does not have any other local or global optima, then this type of functions are called unimodal. On the other hand, if an objective function has more than one optimum which may be local or global, then this kind of functions is called multimodal. Figure 2.1 gives some examples of unimodal and multimodal functions in 2-dimension space.

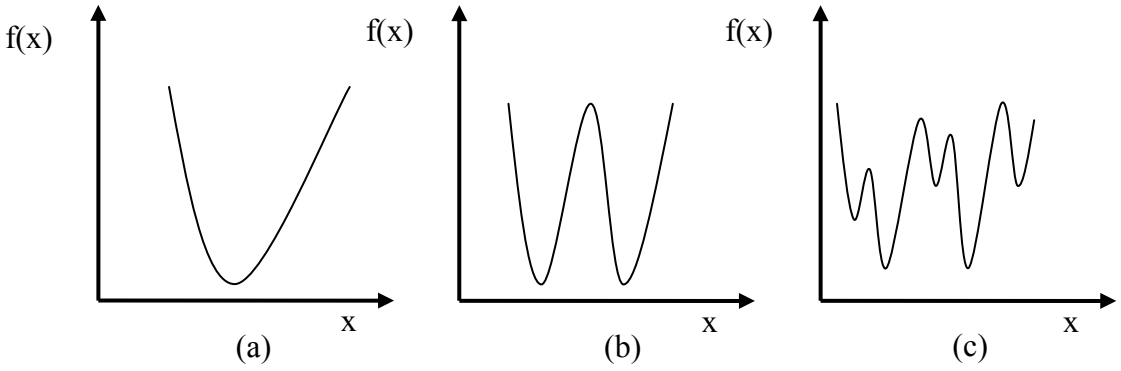


Figure 2.1. (a) Unimodal with one single optimum, (b) Multimodal with two global optima, (c) Multimodal with two global and three local optima.

Most of the real world problems carry multimodal characteristics; hence developing efficient algorithms for multimodal optimization problems is still a research area. Main goal of a multimodal optimization algorithm is to find the global optima of equal quality, if exists more than one [4]. In some cases, even local optima can be of interest as well.

This thesis will focus on multimodal function optimization using Particle Swarm theory. Therefore, all examples and benchmark functions will be multimodal. Since maximization functions can be converted to minimization functions by only changing the sign of the function, all of the optimization functions that are used in this thesis are converted to minimization problem.

2.2. Evolutionary Computation

Evolutionary computation defines a number of computational models for evolutionary processes. Evolutionary Computation (EC) techniques exploit a set of

possible solutions, called population, and detect the optimal problem solution through cooperation and competition between the individuals of the population. An individual's endurance depends on its ability to adapt continuously to the environment to make better use of the resources available. The following factors may affect the evolution in an environment:

- Environmental changes, such as changes in the quantity of resources
- Communication between the members of the population
- Influences of other populations

Depending on the type of the evolutionary algorithm, a population of individuals is adapted using evolutionary operators and strategies in an attempt to find an optimal solution. These evolutionary operators influence individuals to produce better individuals. To simulate the evolutionary process, individuals are evaluated at regular intervals using a fitness function which determines the quality of the individual.

The most commonly used population-based EC techniques, such as Evolution Strategies (ES), Genetic Algorithms(GA), Genetic Programming (GP), Evolutionary Programming (EP) and Swarm Intelligence (SI) are inspired from the evolution of nature [5][6].

2.3. Multimodal Optimization Using Evolutionary Algorithms

Several evolutionary algorithms are proposed for solving multimodal problems. These algorithms can be categorized into two methods; iterative and subpopulation [7].

In iterative methods, the algorithm is applied several times consecutively to locate an optimum during each run. In a sense, each optimum is located one at a time. Iterative methods use various techniques to prevent the optimization method from searching of the search space that have already been explored. Tabu Search [8] and the Sequential Niche Technique [9] fall into this category.

In subpopulation methods, as the name suggests, the population is divided into

subpopulations to search optima simultaneously. If the subpopulations are not allowed to communicate with each other, then the method turns out to be an iterative one. Therefore, parallel subpopulation methods use some communication between the subpopulations to allow "good characteristics" of individuals to spread.

To create and maintain these subpopulations some diversity maintenance methods are used. These diversity maintenance methods can also be subdivided into two [10]:

- Implicit approaches
  i. Geographical separation, such as *island model* and *diffusion model*,
  ii. Speciation, such as *automatic speciation*, *species* and *niching*.
- Explicit approaches
  i. Compete for resources, such as *fitness sharing*, *clearing* and *objective function stretching*,
  ii. Compete for each other for survival, such as *crowding*.

2.3.1. Island Model

In island models [11], evolution occurs in multiple parallel subpopulations. Each of them runs a local EA, evolves independently with rare migrations of selected individuals among subpopulations.

2.3.2. Diffusion Model

Diffusion models [12] assign one individual per processor. The local neighborhood topology is assumed and individuals are allowed to mate only within the neighborhood, called a deme. The demes overlap by an amount that depends on their shape and size and in this way create an implicit migration mechanism. Each processor runs identical EA which select parents from the local neighborhood, produces an offspring, and decides whether to replace the current individual with an offspring.

2.3.3. Automatic Speciation

Automatic speciation [13] restricts the mating of the individuals according to their

similarities. This similarity may be phenotypic or genotypic. That is, only individuals who are phenotypically or genotypically similar can be mate. One other method of automatic speciation is done by adding one bit to each individual. These bits are randomly initialized in the initialization phase, and then only similar individuals according to their new bits are mated.

2.3.4. Species

A species is defined as a group of individuals with similar biological features capable of interbreeding among themselves but that are unable to breed with individuals outside their group. In EA, species can be defined as similar individuals in terms of similarity metrics. By dividing the population into species, usually by mating individuals that are close to each other in the search space, evolutionary algorithms performs better on multimodal problems [7]. By forbidding or limiting interaction between individuals in different species, each species will generally converge on its own optimum. This limitation brings a homogenous phenotypic structure while causing differences between species. At the end of the run, the best individual of each species is considered to represent the location of the species' optimum.

2.3.5. Niching

Niching methods maintain population diversity and permit the algorithm to investigate many peaks in parallel [14]. On the other hand, they prevent the algorithms from being trapped in local optima of the search space. Niching algorithms are based on the mechanics of natural ecosystems. In nature, animals compete to survive by hunting, feeding, breeding, etc., and different species evolve to fill each role. A niche can be viewed as a subspace in the environment that can support different types of life.

For each niche, the physical resources are finite and must be shared among the population of that niche. By analogy, niching methods tend to achieve a natural emergence of niches and species in the environment. A niche is commonly referred to as an optimum of the domain, the fitness representing the resources of that niche.

2.3.6. Fitness Sharing

The fitness sharing method is probably the best known and also used among niching techniques. It was originally introduced in [15] and improved in [16]. The algorithm regards each niche as a finite resource, and shares this resource among all individuals in the niche. Individuals are encouraged to populate a particular area of the search space by adapting their fitness based on the number of other individuals that populate the same area. It lowers each population element's fitness by an amount nearly equal to the number of similar individuals in the population. For a maximization problem the fitness f_i of individual i is adapted to its shared fitness as in Equation (2.1).

$$f_i' = \frac{f_i}{\sum_{j=1}^{N} sh(d_{ij})} \tag{2.1}$$

The sharing function sh is defined in Equation (2.2).

$$sh(d) = \begin{cases} 1-(d/\sigma_{share})^{\alpha} & \text{if } d < \sigma_{share} \\ 0 & \text{otherwise} \end{cases} \tag{2.2}$$

The symbol d_{ij} represents the distance between the individuals i and j. So, if two individual come close to each other less than σ_{share} then their original fitness is decreased. Genotypic or phenotypic distance functions can be used. This approach punishes the individuals that come close to each other. So, they produce some niches along the search space. The parameter σ_{share} denotes the niche radius. This method requires the prior knowledge of the distances between optima for the setting of σ_{share}.

2.3.7. Objective Function Stretching

The objective function stretching was applied as a sequential niching technique by Parsopoulos and Vrahatis [17]. This technique changes the form of an optimization problem's fitness function to remove local minima. When a solution is detected during the

search process, the stretching operator is applied to remove the detected solution from the search space. Ongoing iterations of the searching algorithm can then focus on the other parts of the search space. The detailed explanation of the algorithm is given in Section 3.5.

2.3.8. Crowding

Crowding, introduced by De Jong [18], was originally developed as a diversity maintenance technique. It is inspired by a naturally occurring phenomenon in ecologies, namely competition amongst similar individuals for limited resources. Similar individuals compete to occupy the same ecological niche, while dissimilar individuals do not compete, as they no not occupy the same ecological niche. When a niche has reached its carrying capacity, that is, if it is being occupied by the maximum number of individuals that can exist within it, older individuals are replaced by newer individuals. The carrying capacity of the niche does not change, so population size remains constant. Some improvements have been done to standard crowding algorithm in Deterministic Crowding [19] and Restricted Tournament Selection [20].

2.3.9. Clearing

The clearing method is very similar to fitness sharing but is based on the concept of limited resources of the environment [21]. Instead of sharing the resources between all individuals of a single subpopulation as in fitness sharing, clearing supplies them only to the best members of the subpopulation. In practice, the capacity of a niche specifies the maximum number of elements that this niche can accept. Thus, clearing preserves the fitness of the best individuals, called dominant individuals, of the niche and resets the fitness of the others that belong to the same subpopulation, called dominated individuals. As in the sharing method, individuals belong to the same niche if their distance in the search space is less than a clearing radius. Clearing can be coupled with elitism strategies to preserve the best elements of the niches during the generations. The order of complexity of the basic clearing procedure is where the number of niches maintained during the search.

2.4. Swarm Intelligence

Swarm intelligence is a new evolutionary computation model combining concepts and principles derived from several fields, such as biology, mathematics and social sciences. The basic idea underlying swarm intelligence is that simple individuals interact to produce a complex intelligent behavior.

A swarm has been defined as a set of individuals which communicate directly or indirectly with each other, to collectively carry out a distributed problem solving. Although there is normally no centralized control structure stating how individuals should behave, local interactions between them often lead to the appearance of global behavior. Examples of systems like this can be found in nature, including ant colonies, bird flocking, bacteria molding and fish schooling.

A high-level view of a swarm suggests that the *n* agents in the swarm are cooperating to achieve some target. This "collective intelligence" seems to emerge from what are often large groups of relatively simple agents. The agents use simple local rules to manage their actions and via the interactions of the entire group, the swarm achieves its aim. A type of "self-organization" emerges from the actions of the group.

Swarm intelligence is the evolving collective intelligence of groups of simple autonomous agents. Here, an autonomous agent is a subsystem that interacts with its environment, which probably consists of other agents, but acts relatively independently from all other agents. The autonomous agent does not follow commands from a leader, or some global plan [22]. For example, for a bird to join in a flock, it only adjusts its movements to coordinate with the movements of its flock mates, typically its "neighbors" that are close to it in the flock. A bird in a flock simply tries to stay close to its neighbors, but avoid collisions with them. Each bird does not take commands from any leader bird since there is no lead bird. Any bird can fly in the front, center and back of the swarm. Swarm behavior helps birds take advantage of several things including protection from predators, and searching for food.

The emergent intelligent behavior derives primarily from two principles: self-

organization and stigmergy (stimulation by work).

From a very abstract perspective self organization relies on four basic ingredients:

- Activity amplification by positive feedback.
- Activity balancing by negative feedback.
- Amplification of random fluctuations; randomness is crucial to discovery of new solutions.
- Interaction among multiple agents. Usually agents utilize results of their own activities as well as others.

Stigmergy, or indirect communication through the environment, is the other primary principle behind swarm intelligence. Stigmergy is based on:

- Work as behavioral response to the environmental state.
- An environment that serves as a work state memory.
- Work that does not depend on specific agents.

Swarm intelligence boasts a number of advantages due to the use of mobile agents and stigmergy. These are:

- Scalability: Population of the agents can be adapted according to the problem size. Scalability is also promoted by local and distributed agent interactions.
- Fault tolerance: Swarm intelligent processes do not rely on a centralized control mechanism. Therefore the loss of a few agents does not result in catastrophic failure, but rather leads to graceful, scalable degradation.
- Adaptation: Agents can change, die or reproduce, according to system changes.
- Autonomy: Little or no human supervision is required.
- Parallelism: Agent's operations are inherently parallel.

Two interesting swarm intelligence techniques currently in existence are Ant Colony Optimization (ACO) and Particle Swarm Optimization (PSO).

ACO is a metaheuristic optimization algorithm that can be used to find approximate solutions to difficult combinatorial optimization problems. In ACO artificial ants build

solutions by moving on the problem graph and they deposit artificial pheromone on the graph in such a way that future artificial ants can build better solutions. ACO has been successfully applied to an impressive number of optimization problems.

PSO is a global optimization technique for dealing with problems in which a best solution can be represented as a point or surface in an n-dimensional space. Hypotheses are plotted in this space and seeded with an initial velocity, as well as a communication channel between the particles. Particles then move through the solution space, and are evaluated according to some fitness criterion after each time step. Over time, particles are accelerated towards those particles within their communication grouping which have better fitness values. The main advantage of such an approach over other global optimization strategies such as simulated annealing is that the large numbers of members that make up the particle swarm make the technique impressively resilient to the problem of local optima.

2.5. Particle Swarm Optimization

In 1995, Eberhart and Kennedy proposed the Particle Swarm Optimization (PSO) algorithm for solving optimization problems [23]. This stochastic evolutionary computation technique is based on the movement and intelligence of swarms looking for the most favorable feeding area. The members of the swarm, particles, move along the search space depending on self experience and social interactions during the search. The movement, the velocity of each particle depends on self and its neighbors' experience. So, each particle adapts its position based on previously visited best positions by itself and its neighbors. A particle in a swarm starts its flight from a random location with a random velocity, and at each step adjusts its velocity to move to another location based on the Equations (2.3) and (2.4).

$$v_{id}(t+1)=w\, v_{id}(t)+c_1\, r_1(t)\, (y_{id}(t)-x_{id}(t))+c_2\, r_2(t)\, (y_{gd}(t)-x_{id}(t)) \,, \tag{2.3}$$

$$x_{id}(t+1)= x_{id}(t)+ v_{id}(t+1) \,, \tag{2.4}$$

In Equations (2.3) and (2.4), $x_{id}(t)$ is the position of the i^{th} particle at time t on dimension d, v is velocity, w is the inertia weight, c_1 and c_2 are acceleration constants, r_1 and r_2 are uniform random numbers in [0,1], γ_i is the i^{th} particle's best position that has been found so far, and y_g is the best position visited by the neighbors. Generally, neighborhood is chosen as the whole population for global optimization.

The Equation (2.3) consists of three main parts: The inertia of pervious velocity; the "cognition" part; and the "social" part. The "cognition" part is the particle's private thinking, and the "social" part is the cooperation among the neighbors [24]. In traditional evolution, latter two represent the local stress and regional history [25]. Equation (2.5) and (2.6) defines the velocity update equations of the cognition only model and social only model, respectively.

$$v_{id}(t+1)=w\, v_{id}(t)+c_1\, r_1(t)\, (\gamma_{id}(t)-x_{id}(t)) \tag{2.5}$$

$$v_{id}(t+1)=w\, v_{id}(t)+c_2\, r_2(t)\, (y_{gd}(t)-x_{id}(t))\ , \tag{2.6}$$

1. Initialize particles positions and velocities,
2. Evaluate the fitness of each particle,
3. Set the current position as the best position for each particle,
4. Determine the global best particle,
5. While stopping criteria are not met, for each particle do
    a. Update the position,
    b. Calculate fitness,
    c. If it is better than particle's best, then set it to the particle's best position,
    d. If it is better than the global best, then set it to the global best position,
    e. Update velocity.

Figure 2.2. The PSO algorithm

Another concept in PSO is the parameter tuning. The algorithm has a few parameters:

number of particles n, inertia weight w, acceleration constants c_1 and c_2, maximum velocity V_{max}.

If the number of particles is increased, then while gathering improved exploration ability, computational cost is raised. Therefore, speed of the algorithm will fall. The main rule for determining the size of the swarm is: Increase the size of the swarm when the complexity of the search space increases.

The inertia weight determines the effect of the current velocity when calculating the new velocity. If the inertia weight is greater than one, then velocity of the particles will increase on each epoch, causing to explore larger regions. If it is less than one, then velocity of the particles will decrease on each epoch, causing to make finer search near current position [26]. There are many algorithms for the calculation of the inertia weight. Most of them decrease the value over time [27].

Generally, velocity of the particles is limited by the constant V_{max}, to improve the resolution of the search by preventing particles from moving too rapidly through the search space [28].

Acceleration constants determine the influence of the personal and swarm's experience. Cognitive (c_1) and social (c_2) acceleration constants bias the particle to either move more towards its previous best position, or its newfound neighborhood best position.

Two main neighborhood information sharing structures exist [29]:

- Global Best (*gbest*): The whole swarm constitutes the neighborhood of a particle. That is, all of the particles are interconnected with other particles in the swarm. When a new global best position is found, this information is immediately shared with all other particles. So, all of them use the same global best position on calculation of their velocity. The entire swarm moves towards the global best position. This may result in premature convergence on suboptimal solutions.

- Local Best (*lbest*): In the *lbest* structure, neighborhood of a particle is determined

according to their indices in the swarm. If the neighborhood size is "2*m*+1" then, *m* left particles and *m* right particles constructs the neighborhood. For example, if the neighborhood size is 3, particles which have indices *i*-1 and *i*+1 are the neighbors of particle *i*. Particle *i* uses the best position of this three particles as y_g in Equation (2.1) on the calculation of its velocity. Since the neighbors of particles intersect, a new found the best value may be shared with other particles as the time passes. There is no global best particle, but rather a local best particle for each neighborhood. Each particle is therefore only influenced by its neighbors, and not the whole swarm.

The *gbest* algorithm can be seen as the *lbest* algorithm with one neighborhood consisting of all the particles. The size of neighborhoods plays an important role in the *lbest* algorithm. It has been shown that smaller neighborhoods result in slower convergence, but generally lead to better results, since a larger part of the search space is explored and particles are less likely to be trapped in local optima [27].

Some research has been done on the parameter selection concept. For more details on how to choose PSO parameters refer to [25][30][31].

2.5.1. The Guaranteed Convergence PSO (GCPSO)

When all of the particles catch up the best particle before finding the optimum, then current position, the best position and the global best position will be the same for all particles. This will cause the second and third part of the velocity update equation (Equation (2.3)) to be zero. Consequently, only the first part, inertia weight, will help the particles to move to a better position. If the inertia weight is chosen to be less than 1, which is suggested, it will possibly make the velocity zero. That is, in standard PSO, all particles may stop moving before finding an optimum. Van den Bergh and Engelbrecht introduced a new method for PSO to prevent this premature convergence behaviour [32]: Guaranteed Convergence PSO (GCPSO). One of the multimodal adaptations of PSO, the Niching PSO, uses GCPSO to avoid premature convergence. The details of the Niching PSO is given in Section 3.2.

In GCPSO, in order to keep the global best particle moving, a new velocity update equation is introduced only for the best particle g as in Equation (2.7).

$$v_{gd}(t+1) = -x_{gd}(t) + y_{gd}(t) + w\, v_{gd}(t) + \rho(t)\,(1- 2r_d(t)) \tag{2.7}$$

All of the other particles in the swarm continue using the standard PSO velocity update equation given Equation (2.3). In Equation (2.7), the first and second parts resets the position of the particle to its best position. Then with the third part, a vector representing the current search direction is added. The last term generates a random value in $[-\rho(t),\rho(t)]$. With this term, the global best particle will make a random search near global best position. The diameter of this random search area is controlled by the parameter ρ. The new parameter ρ is the scaling factor and calculated using the formula given in Equation (2.8).

$$\rho(t+1) = \begin{cases} 2\rho(t), & \text{if } \#\text{successes} > s_c \\ 0.5\rho(t), & \text{if } \#\text{failures} > f_c \\ \rho(t), & \text{otherwise} \end{cases} \tag{2.8}$$

In Equation (2.8), *#successes* and *#failures* are the number of consecutive successes and failures respectively. A failure occurs when there is no improvement on the value of global best fitness between consecutive iterations. That is, there is a failure if $f(y_g(t+1))=f(y_g(t))$. Thus, there is a success if $f(y_g(t+1))<f(y_g(t))$ for a minimization function. When a failure occurs, the value of *#successes* is reset to 0, similarly, when a success occurs, the value of *#failures* is reset to 0. More formally, the Equation (2.9) and (2.10) should be implemented.

$$\#successes(t+1) > \#successes(t) \;\rightarrow\; \#failures(t+1) = 0 \tag{2.9}$$

$$\#failures(t+1) > \#failures(t) \;\rightarrow\; \#successes(t+1) = 0 \tag{2.10}$$

The scailing factor ρ is initially assigned to 1, $\rho(0)=1.0$. The threshold parameters success count, *sc*, and failure count, *fc*, depend on the objective function. If the number of

dimension is high, it will be difficult to obtain better values using a random search in only a few iterations, so it is recommended to set $fc = 5$, $sc = 15$. With these settings, the algorithm will be quicker to punish a poor ρ setting than it is to reward a successful ρ value.

The detailed discussion of these parameters and different neigborhood topologies can be found in [32] and [33].

3. PSO FOR MULTIMODAL OPTIMIZATION

The uniqueness of PSO's ability in adaptively adjusting particles' positions based on the dynamic interactions with other particles in the population makes it well suited for handling multimodal optimization problems. If suitable particles can be determined as the appropriate neighborhood best particles to guide different portions of the swarm population moving towards different optima, then essentially we will be able to use a PSO to optimize over a multimodal fitness landscape. Ideally multiple optima will be found. Now the question is how to determine which particles would be suitable as neighborhood bests; and how to assign them to the suitable particles in the population so that they will move towards different optima accordingly.

One approach to combat this problem is to allow the population to search for multiple optima (either global or local) simultaneously. Striving to locate multiple optima has two advantages. Firstly, by locating multiple optima, the likelihood of finding the global optimum is increased; secondly, when dealing with real-world problems, for some practical reasons, it is often desirable for the designer to choose from a diverse set of good solutions, which may be equally good global optima or even second best optima.

Scientists developed several algorithms for solving multimodal problems using PSO. In this thesis, the following algorithms are analyzed, implemented and compared: Species-based PSO, Niching PSO, nbest PSO, Unified PSO and Stretching PSO. A new algorithm called PSO with Craziness is also introduced.

3.1. Species Based PSO (SPSO)

After gathering good results with their species conserving genetic algorithm (SCGA) for multimodal optimization [7], Xiaodong Li saw that the technique of dividing the population based on the notion of species can be applied to Particle Swarm.So, he introduced a new algorithm for solving multimodal problems using particle swarm called Species-based particle swarm optimization (SPSO) [34]. Species-based PSO algorithm divides the swarm into subswarms called species, according to their similarity (Euclidean

distance) and a parameter called species radius r_s. In each species, the best fit particle is the species seed, and the boundary of the species is the circular area with in radius r_s according to this seed. At each iteration, particles in the entire swarm move according to the species they belong to. Afterwards, particles are evaluated and species are redefined. The swarm is allowed to achieve the multiple optima in a parallel manner. Convergence rate of the algorithm is enhanced by two sub-processes: communication of the particles in the swarm through the PSO algorithm and the reconstruction of the species.

1. Initialize particles
2. Update fitness of each particle.
3. Calculate L_{sorted} = containing all particles sorted in decreasing order fitness
4. Calculate S containing dominating particles identified as species seeds as :

```
S = Φ;
For each s Є L_sorted do
        found ← FALSE;
        for all p Є S do
                if  d(s,p) ≤ r_s then  //d(x,y) is the Euclidean dist. fucn.
                        found ← TRUE;
                        break;
                end
        end
        if (not found) then
                let S ← s U {S}
        end
end
```

5. Train particles one iteration using species seed as the best member.
6. Repeat from 2 until stopping criteria are met.

Figure 3.1. The Species-Based PSO (SPSO) algorithm

The species seed identification algorithm is adapted from [7]. With this algorithm, at each iteration step, different numbers of species are generated and different particels can be

the seed of a species. So, on each epoch, each particle may use different particle's best position as the *lbest*. Figure 3.1 summarizes the algorithm.

After randomly initializing and calculating the fitness of each particle, species identification algorithm starts. This algorithm requires an input array called L_{sorted}, which contains the indices of all particles sorted in decreasing order of fitness. The aim is to identify the species seeds and the species that all particles belong to. So, the set containing species seed S is set to Φ initially. Then, all particles in L_{sorted} are controlled (from best to least-fit) if they are close to a seed in S within the distance r_s. If a seed is found, then particle chooses this seed as its species seed. If it can not found a species to join, a new species is generated by adding this particle to S. Finally, after finding the species it belongs to, each particle adjusts its position using the species seed as the newly identified neighbourhood best (*lbest*).

This speciation provides particles within the same species to move to positions that may make them better. Additionally, since species are formed around different optima, this algorithm can found multiple optima in parallel. Figure 3.2 shows how different species move towards different optima in parallel.

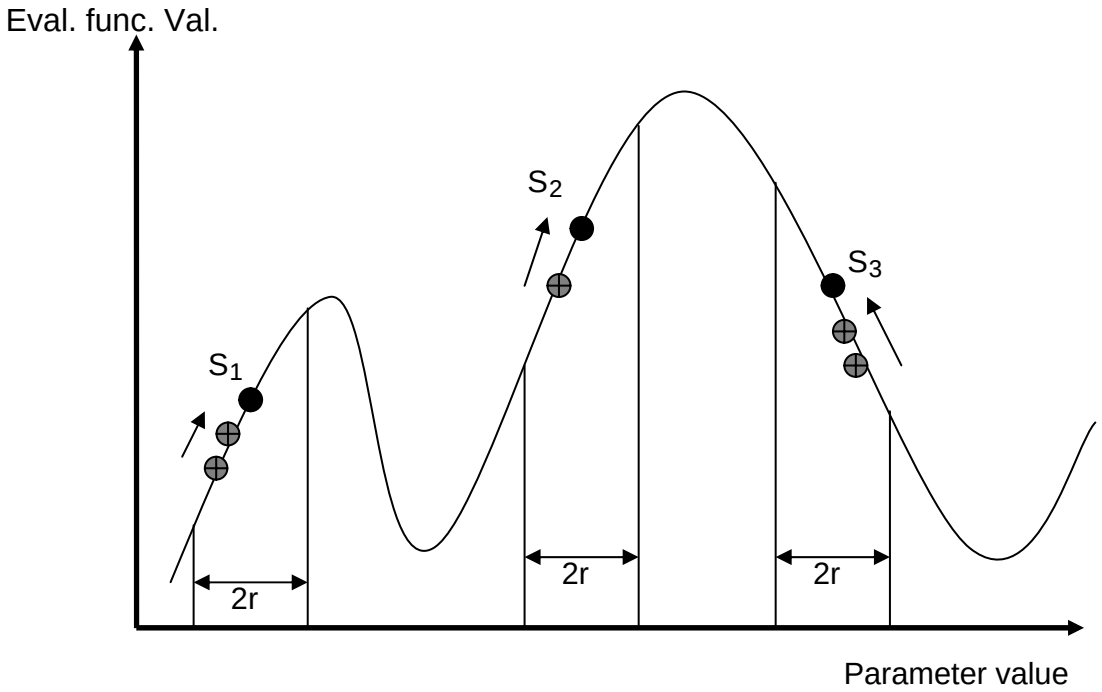


Figure 3.2. Species moving towards optima in parallel

If enough number of particles is used, there is no need to pre-specify the total number of optima. This speciation also allows dynamically finding all of them.

3.2. The Niching Particle Swarm Optimizer (NichePSO)

R.Brits, A.P. Engelbrecht and F. van den Bergh [35] proposed an algorithm to find multiple optima in parallel using particle swarm algorithm, called NichePSO. Adapting the algorithms of Løvbjerg *et al.* [36], in which subswarms are used for improving swarm diversity to find a good quality single global optimum, NichePSO algorithm uses subswarms for optimizing niches in the objective function space.

In the first stage of the algorithm, the particles are distributed uniformly along search space. This initial swarm is referred as the main swarm. When iterations start and particles move towards the better fitness area, monitoring of the particles' fitness is also started. If a particle's fitness is not changed significantly for some epoch, its position is set to be a potential solution. Then, this particle is removed from the main swarm and a new subswarm is generated. The main swarm loses its members as the new subswarms created. So, these dynamically generated subswarms will find different number of optima in parallel. The following sections describe the steps of the algorithm in more detail.

3.2.1. Initialization

To increase the success rate of the Niche PSO by means of a uniform initial distribution of particles throughout the search space, scientists recommended Faure-sequences [37] for generating initial particle positions.

3.2.2. Main Swarm Training

Since the aim of the multimodal algorithms is not only finding global optima but some local optima, particles in the main swarm uses cognition only model to calculate their velocity. Consequently, each particle in the main swarm makes a local search based on its own experience without getting any impression from other particles.

3.2.3. Identification of Niches

Different from Stretching PSO, NichePSO uses a fitness monitoring technique to identify niches. That is, if the standard deviation in particle *i*'s fitness, σ_i, for some iteration e_σ, is below a predefined value, δ, then a subswarm may be created with particle *i* and its closest neighbor. The 'closest neighbor' is defined as the closest particle in the main swarm to particle *i*. With this approach, local optima can be found when $\sigma_i < \delta$. If the local optima are undesired, the fitness's of particles can be compared with a given threshold fitness value.

The following sections help the algorithm not to perform duplicated searches on the same niches.

3.2.4. Absorption of Particles into a Subswarm

Since a particle in the main swarm does not have any information of the niches where the subswarms are, there is a possibility that some particles in the main swarm and some subswarms may exist in the same niche. So, particles in the main swarm are absorbed by a subswarm if they move into it. That is, a particle *i* will be absorbed by a subswarm S_j when

$$\| x_i - y_{gS_j} \| < R_j \tag{3.1}$$

where *Rj*, the radius of subswarm *Sj*, is defined as

$$R_j = \max \{ \| y_{gS_j} - x_{S_{j,i}} \| \} \tag{3.2}$$

where $x_{S_{j,i}}$ represents all particles in S_j, y_{gS_j} is the global best particle in S_j.

This absorption is done based on the following suppositions:

- If a particle in the main swarm is searching the same niche with a subswarm, it will increase the diversity of the subswarm when they are merged. Consequently,

this increased diversity will help the subswarm to locate a better solution more rapidly.

- A particle's performance of finding good solution is increased with social impact. That is, success rate of a single particle which is influenced by only its experience is worse than the success rate of a particle which is influenced by its and its neighbor's experience.

3.2.5. Merging Subswarms

With the similar reasons given in the previous section, subswarms may be merged when they are attempting to optimize the same solution. That is, subswarms are merged when they *intersect*. The intersection occurs if the following inequality is satisfied:

$$\| y_{gS_{j1}} - y_{gS_{j2}} \| < (R_{j1} + R_{j2}) \tag{3.3}$$

where, $y_{gS_{j1}}$ and $y_{gS_{j2}}$ are the best particles in subswarms S_{j1} and S_{j2}; R_{j1} and R_{j2} are the subswarm radius of S_{j1} and S_{j2}, respectively.

To prevent the problem of unconsciousness for two subswarms are in the same niche, when $R_{j1}= R_{j2}=0$, a new parameter, μ, is introduced. With this parameter, two subswarms are merged when Equation (3.3) holds or the Equation (3.4) holds.

$$\| y_{gS_{j1}} - y_{gS_{j2}} \| < \mu \tag{3.4}$$

To ensure the similarity of the subswarms, that are going to be merged, parameter μ should be a small number as 0.001. Different values of μ are tested by the scientists and it has been found that NichePSO performs better when $\mu < 0.1$.

In summary, when Equation (3.3) or (3.4) is true for two subswarms S_{j1} and S_{j2}, then a new subswarm, which includes the all particles in subswarms S_{j1} and S_{j2}, is generated. S_{j1} and S_{j2} are removed from the subswarm list.

3.2.6. Subswarm Training

When a subswarm is created, it only consists of two particles at most. As described in Section 2.5.1, such small swarms may get stuck into a suboptimal position. So, GCPSO algorithm, defined in Section 2.5.1, is used for subswarm training to prevent this stagnation. Figure 3.3 shows the summary of the NichePSO algorithm.

1. Initialize main particle swarm
2. Train main swarm particles using one iteration of the cognition only model.
3. Update fitness of each main swarm particle.
4. For each subswarm:
    a. Train subswarm particles using one iteration of the GCPSO algorithm.
    b. Update each particle's fitness.
    c. Update swarm radius
5. If possible, merge subswarms
6. Allow subswarms to absorb any particles from the main swarm that moved into it.
7. Search main swarm for any particle that meets the partitioning criteria – If any is found create a new subswarm with this particle and its closest neighbor.
8. Repeat from 2 until stopping criteria are met.

Figure 3.3. The NichePSO algorithm

3.3. Modified Niching PSO (mNichePSO)

During the experiments, it has been observed that the niche radius of a subswarm may increase, spanning the whole search space and causing most of the particles to converge to a single optimum. Hence, a modified version, referred as *mNichePSO* is proposed to prevent this type of behavior. Simply, two subswarms are merged when Equation (3.3) or (3.4) holds, and Equation (3.5) holds.

$$R_{j1}+R_{j2} <= R_{max} \tag{3.5}$$

That is, two subswarms are not merged when they are not close enough. This closeness is defined by the parameter maximum niche radius, R_{max}. So, with this parameter, two different subswarms that are on different niches will not merge. Consequently, they will found two different optima.

Experiments given in Section 4 shown that, this modified version of NichePSO performs better that the original Niche PSO.

3.4. The *nbest* Particle Swarm Optimizer

R.Brits, A.P. Engelbrecht and F. van den Bergh [14] introduced a modified version of particle swarm optimizer called *nbest* PSO to solve systems of unconstrained equations which may have multiple solutions in search space. However, experiments given in Section 4 shown that, this algorithm can also be used on multimodal function optimization.

To increase diversity while sharing information, *nbest* PSO introduces a new definition for the neighborhood best position. For each particle i, k closest particles are found, and the neighborhood best position y_{gi} is calculated as the center of mass of these positions. To put into formula, if the set B_i contains the k closest particles to i, at any timestamp t, then y_{gi} is calculated as:

$$y_{gi}(t) = \frac{1}{k} \sum_{h=1}^{k} B_{ih}(t) \tag{3.6}$$

Where $B_{ih}(t)$ is the h^{th} particle in B_i at time t. The fitness values of these particles are not taken into account, that is, all k particles has the same equal effect. After calculating the neighborhood best position y_{gi} for each particle, the same velocity update equation in Equation (2.3) is used.

As it can be seen from Equation (3.6), if the value of k is equal to swarm size, the neighborhood of a particle will be the whole swam. Consequently, diversity of the swarm and capability of finding multiple optima will decrease. Linearly and exponentially decreasing and constant k values are analyzed by the researchers, and they found that,

linearly decreasing k value yields the best performance. Constant values of k greater than one provided some convergence, but if k equals to one, no convergence observed.

3.5. The Stretched PSO

Parsopoulos et al. introduced the Stretched Particle Swarm Optimizer (SPSO) [38], for the alleviation of the local minima problem. In addition, they observed that, this method can also be used on multimodal functions to locate all global optima [17].

In order to solve the problems of particle swarm optimization on multimodal functions, in Stretched PSO, objective function is stretched on a found solution which is better than a predefined threshold ϵ. That is, when a particle's fitness is smaller than ϵ, this particle is isolated from the main swarm. Then, the objective function is stretched at the particle's current position in order to prevent the other particles in the main swarm searching on the same area. On subsequent iterations, main swarm will focus on the area different than this position and its neighbor. After the isolation of that particle, a new randomly generated particle is added to the main swarm instead of it.

The stretching operation consists of a two-stage transformation of the original function $f(x)$. If a local optimum is found on x^*, that is, for a minimization problem, if $f(x^*)<\epsilon$ then for all remaining particles the fitness function is redefined as in Equation (3.8). The first transformation in Equation (3.7) stretches the objective function so that, local minima worse than $f(x^*)$ are disappeared and all positions where $f(x)$ is better than $f(x^*)$ are left unaffected. The second transformation function in Equation (3.8) constructs a peak on the neighborhood of x^*, that is, for a minimization problem, the neighborhood of x^* is elevated. The sign function used in Equation (3.8) is defined as in Equation (3.9).

$$G(x)= f(x) + \gamma_1 \frac{\left\|x-x^*\right\|(sign(f(x)-f(x^*))+1)}{2} \tag{3.7}$$

$$H(x)= G(x) + \gamma_2 \frac{(sign(f(x)-f(x^*))+1)}{2\tanh(\mu(G(x)-G(x^*)))} \tag{3.8}$$

For example, the stages of the stretching technique is applied and shown in Figures 3.4, 3.5 and 3.6 for the function in Equation (3.10).

$$sign(x)=\begin{cases}+1, & x>0,\\ 0, & x=0,\\ -1, & x<0\end{cases} \tag{3.9}$$

$$f(x)=1\text{-}\sin(5\pi x)^6 \tag{3.10}$$

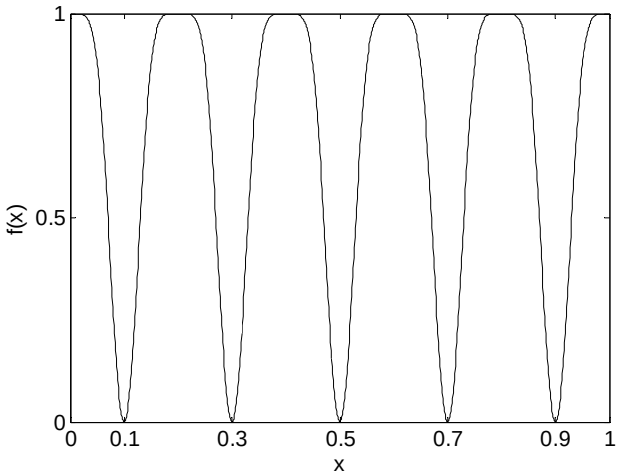


Figure 3.4. The plot of function given in Equation (3.10)

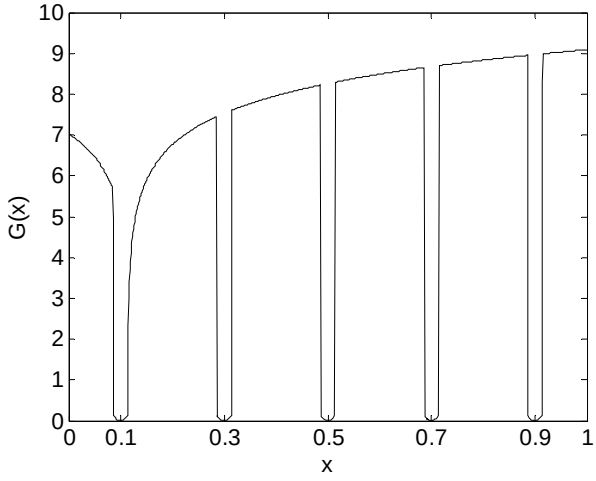


Figure 3.5. The first stage ($G(x)$) of function given in Equation (3.10)

This function has 5 equal fitness minima with value of 0 at 0.1, 0.3, 0.5, 0.7 and 0.9. The value of x^* is chosen as 0.114 and $f(x^*)$ is 0.1361. The parameters γ_1, γ_2 and μ are set to 10^4, 1.0 and 10^{-10} respectively.

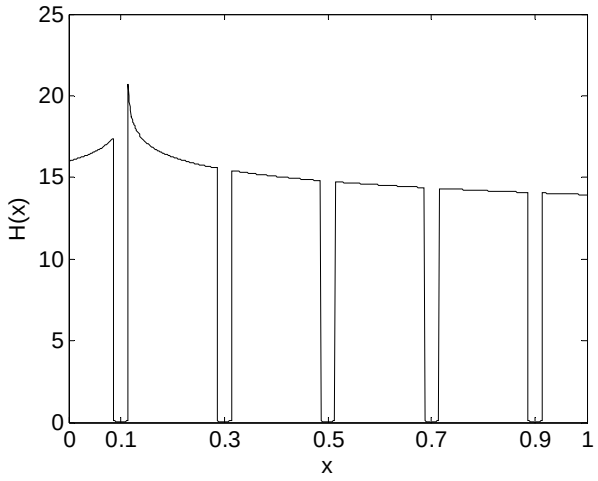


Figure 3.6. The second stage ($H(x)$) of function given in Equation (3.10).

1. Initialize particles
2. Set local minima counter LMC=0
3. Train particles one iteration.
4. Update fitness.
5. If a local minimum has been detected, then
   i. Set LMC=LMC+1,
   ii. Set $f(x)$=$H(x)$,
   iii. Isolate the particle and create a small swarm in its small neigborhood,
   iv. Add a new randomly generated particle to the main swarm.
6. Repeat from 3 until stopping criteria are met.

Figure 3.7. The Stretched PSO algorithm

In the first stage, all positions which have functional values greater than $f(x^*)$ are stretched, as in Figure 3.5. In the second stage, a peak is constructed in the neighborhood

of x^* (Figure 3.6). Thus, the area around x^* is marked as undesirable. The Stretched PSO can be summarized as in Figure 3.7.

Brits *et al.* [14] observed the following problems in Stretched PSO.

- The stretching done by $G(x)$ makes some steep trenches around the remaining solutions, while producing smooth horizontal formation on the other parts of the search space. This smooth formation prevents other particles to find other optima during the rest of the optimization process. If the width of the trench is very small as in Figure 3.5, the probability of finding these trenches will be very low. This brings a poor performance on finding multiple optima.
- The transformation made by $H(x)$ may cause disappearing of the other optima which are very close to x^*. That is, if more than one desired optima are very close to each other, $H(x)$ may make them undesirable by elevating. For example, if there were an optimum near 0.1 in Figure 3.6, $H(x)$ would remove it.
- The stretching may create new optima in the search space if the parameters γ_1, γ_2 and μ are not tuned.

Consequently, Brits et al. cannot replicate the results of Parsopoulos. Our experiments also shown that, after finding the second optimum, the width of the trenches near optima will decrease and the hill constructed by elevating the area near the second found optima will be very steep. These properties of the Stretched PSO make it unsuitable for multimodal problems.

For example, after finding an optimum on x_1=0.114 where $f(x_1)$=0.1361, if the optimization process finds a new optimum on x_2=0.702 where $f(x_2)$=0.003, the final plot of the $G(x)$ and $H(x)$ will become as in Figure 3.8 and 3.9. The steep hill near 0.702 and too narrow steep trenches near the other unfound optima can be seen in Figure 3.9. This new form of the original function $f(x)$ makes it very difficult to identify the other unknown optima for the optimization process.

As Brits *et al.*, we couldn't gather the results observed by Parsopoulos, either. As a result, we haven't included the Stretched PSO to our comparison tests.

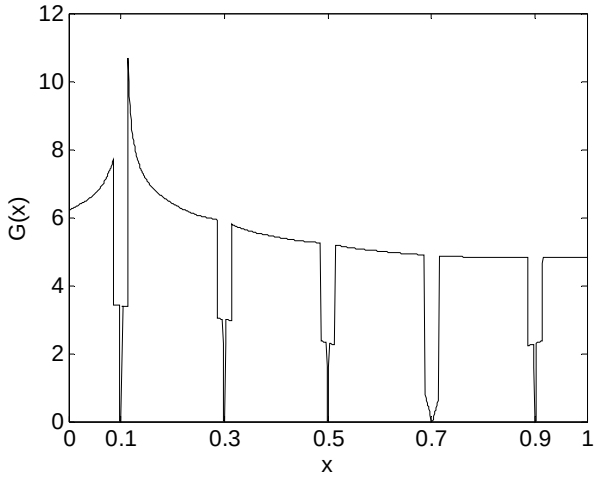


Figure 3.8. The $G(x)$ for the second optimum

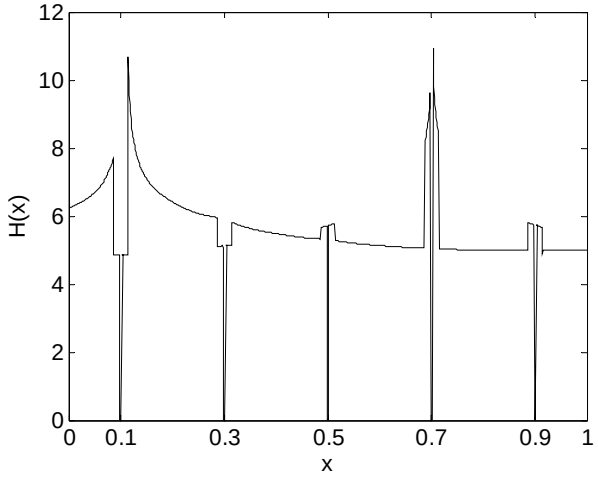


Figure 3.9. The H(x) for the second optimum

3.6. A Unified Particle Swarm Optimization Scheme (UPSO)

The Unified PSO, introduced by K.E. Parsopoulos and M.N. Vrahatis [39], combines the global and local variant of PSO, that is, it combines the exploitation and exploration properties of PSO. Although it is used on unimodal problems [40], it also has the ability on finding multiple optimums of multimodal problems in parallel.

The capability of exploring the whole search space and the capability of exploiting a local optimum of an algorithm improves the performance of a population based optimization algorithm. In addition, a well-arranged balance between these two characteristics also increases its performance. In the standard PSO, all particles are affected by the global best particle. This brings faster convergence with increased exploitation capability. In the *lbest* version of the PSO, information of the best position is passed slowly to the other particles. The speed of this information passing is controlled by the neighborhood size. By bringing the exploration property, this slowness prevents the swarm to get stuck on a single optimum. The neighborhood size determines the balance between exploration and exploitation.

Unified Particle Swarm Optimization scheme introduces a new method to balance the global and local variants of PSO. In this algorithm, velocity update equation is changed and divided into local and global parts. Using Equations (3.11) and (3.12), where $G_i^{(k+1)}$ is the global variant and $L_i^{(k+1)}$ is the local variant, the new velocity update equation for particle x_i is calculated using Equation (3.13).

$$G_i^{(k+1)} = X\,[\, v_i^{(k)} + c_1 r_1 (\mathrm{p_i}^{(k)} - x_i^{(k)}) + c_2 r_2 (p_g^{(k)} - x_i^{(k)})] \,, \tag{3.11}$$

$$L_i^{(k+1)} = X\,[v_i^{(k)} + c_1 r_3\, (p_i^{(k)} - x_i^{(k)}) + c_2 r_4\, (p_{g_i}^{(k)} - x_i^{(k)})] \,, \tag{3.12}$$

$$U_i^{(k+1)} = u\; G_i^{(k+1)} + (1-u)\; L_i^{(k+1)}, \tag{3.13}$$

$$x_i^{(k+1)} = x_i^{(k)} + U_i^{(k+1)} \tag{3.14}$$

where, X is the constriction factor that controls the velocity's magnitude [41], k denotes the iteration number, v is the current velocity, c_1 and c_2 are constants, r_1, r_2, r_3 and r_4 are random numbers, p is the particle's best position, g is the index of the best particle of the whole swarm, g_i is the index of the best particle in the neighborhood of particle i. The new introduced $u \in [0,1]$ is the unification factor determines the effect of the global and local information.

If $u=1$ is chosen, then Equation (3.13) becomes the global PSO, if $u=0$ is chosen then it becomes the local PSO, other values of $u \in (0,1)$ will provide a composition of local and global PSO variant.

Additionally, the Unified PSO is enhanced by a stochastic parameter as shown in Equation (3.15), which is also used to make a balance between local and global parts. Thus, Equation (3.13) can be written as Equation (3.15) which provides a velocity mostly based on the local variant; and Equation (3.16) which provides a velocity mostly based on the global variant. The parameter r_5 is normally distributed in $N(\mu,\sigma^2 I)$.

$$U_i^{(k+1)} = r_5\, u\, G_i^{(k+1)} + (1-u)\, L_i^{(k+1)}, \tag{3.15}$$

$$U_i^{(k+1)} = u\, G_i^{(k+1)} + r_5\, (1-u)\, L_i^{(k+1)} \tag{3.16}$$

If a value close to zero is chosen for the unification factor, local variant will mostly affect the velocity of particles. Consequently, exploration ability of the swarm will increase. Although the Unified PSO is designed to find one single optimum, our experiments shown that, it can find multiple optima in parallel when u=0.1. The detailed results of the experiments are given in Section 4.

3.7. A Parallel Vector-Based Particle Swarm Optimizer (PVPSO)

The Parallel Vector-Based PSO, introduced by I.L. Schoeman and and A.P. Engelbrecht [42], that uses vector operations to find niches in the search space. The early version is the Vector-Based PSO [43], in which niches are searched sequentially. Since the search space may not be symmetric, more than required number of niches may be initially determined. Therefore, the same optima may be searched twice. To avoid this problem Parallel Vector-Based PSO is proposed.

In this algorithm, initial niches are identified similar to the Vector-Based PSO, but all particles are evaluated simultaneously. Velocity update is done using personal best and

neighborhood best positions. Subswarms may be merged and subswarms may absorb other particles that come close to it. The Parallel Vector-Based PSO algorithm can be summarized as in Figure 3.10.

During the experiments, it has been observed that, although particles are moving the same niche but in opposite direction, this algorithm does not realize that they are moving to the same niche. So, smaller subswarms are created and performance of the algorithm gets worse. The detailed results of the experiments are given in Section 4.

1. Initialize particles
2. An initial personal best position for each particle is found by evaluating fitness of a random position near the particle. Then calculate the vector v_{pi} for each particle using Eq.3.17.
3. Niches are identified using the following vector operations:
   i. Set neighbourhood best *gbest* to *pbest* with the best fitness. Calculate v_{gi} using Equation Eq.3.18,
   ii. Calculate the dot product λ for each particle using Equation Eq.3.19,
   iii. Calculate the niche radius – the distance between *gbest* and the nearest particle with a negative dot product,
   iv. Identify all particles inside the niche radius having a positive dot product as belonging to the current niche,
   v. Repeat steps *i* to *iv* for the remaining particles until all particles have been processed.
4. If a niche contains too few particle (less than 3 in this algorithm) new particles are spawned in the vicinity to prevent niches from becoming stationary.
5. Update particles:
   i. For each particle: update its position by using *pbest* and its neighbourhood best *gbest*,
   ii. Update *pbest* and *gbest* if better positions are found, as well as vectors v_{pi} , v_{gi} and their dot product λ. Reapeat this step *k* times,
6. Update the niche radius.
7. Merge niches: If the distance between *gbest* of two niches becomes smaller than ε, particles are merged with the niche where *gbest* has the best fitness. Only particles nearer than ε to *gbest* of that niche will be merged.
8. Repeat steps 5 to 7 *m* times. The total number of iterations of the entire swarm will be *m* k*.

Figure 3.10. The Parallel Vector-Based PSO algorithm

$$v_{pi}(t) = y_i(t) - x_i(t), \tag{3.17}$$

$$v_{gi}(t) = y_g(t) - x_i(t), \tag{3.18}$$

$$\lambda = v_{pi} * v_{gi}, \tag{3.19}$$

3.8. Particle Swarm Optimizer using Craziness and Hill-climbing

Particle Swarm Optimization algorithm based on craziness and hill-climbing (CPSO) enhances the exploration and exploitation capabilities of PSO for solving multimodal problems. To explore several areas in parallel, the main swarm is divided into subswarms according to their geographical positions. Each particle also has a crazy part, which also helps the particle to search different areas. To be more productive on exploiting the promising areas, each particle applies a hill-climbing strategy. The algorithm of CPSO is presented in Figure 3.11.

In this algorithm, the main swarm is divided into subswarms of size n according to their geographical positions in order to determine a set of optima, simultaneously. On every m epochs, subswarms are rearranged according to their current geographical positions. This action provides a type of communication and information diffusion between particles, since a local best value might change within a neighborhood.

Each particle generates two candidate positions, denoted by x^1 and x^2, at each epoch. In Equations (3.17) to (3.22), v^1 and v^2 denote velocities for related candidate positions. In Equation (3.17), y_{gi} is the best position visited so far within the neighborhood of the i^{th} particle. Parameters $maxx_d$ and $minx_d$, in Equation (3.22), are the limits of the search space in dimension d. The first candidate position is computed using Equation (3.20) and (3.21), and the second one is computed using Equation (3.22) during initial moves. For candidate positions x^1 and x^2, each particle makes a decision based on the fitness values. Better position having a better fitness is chosen and the particle makes its move towards it. Due to

the craziness and the hill-climbing components, CPSO improves its exploration and exploitation capabilities, respectively.

1. Initialize particles

2. On the first and every *m* epochs, construct subswarms according to their geographical positions, with neighborhood size *n*.

3. For each particle compute 2 candidate positions:
   a. Use the original PSO, where y_g is the subswarm's best.
   b. if (fitness's of the particle or the particles in its subswarm, did not change for *k* epochs)
      then
         use the original PSO,
         where y_g is the subswarm's best
      else
         generate a random position using Equation 3.22

4. Compute the fitness values of these candidate positions. Choose the position with the better fitness as particles' current position. If the random position produces a better fitness, set the particles' velocity using Equation 3.23.

5. If (for *p* succesive epochs, fitness of the particle did not change)
   then
      reset the velocity of the particle using Equation 3.23.

6. Repeat from 2 until stopping criteria met.

Figure 3.11. The CPSO algorithm

$$v^1_{id}(t+1)=w\, v_{id}(t)+c_1\, r_1(t)\, (y_{id}(t)-x_{id}(t))+c_2\, r_2(t)\, (y_{gid}(t)-x_{id}(t)) \,, \tag{3.20}$$

$$x^1_{id}(t+1)=x_{id}(t)+v^1_{id}(t+1) \,, \tag{3.21}$$

$$x^2_{id}(t+1)= minx_d + r_3\, (maxx_d-minx_d) \,, \tag{3.22}$$

Moving to the better candidate position can be considered as a hill-climbing step; moreover, randomness in candidate positions and velocity resets bring some craziness

factor to the original PSO algorithm. Hence the PSO algorithm is named as PSO using Craziness and Hill-climbing (CPSO). Each particle has two phases: *sampling* and *acceptance*. As a sampling technique either a random walk or the PSO algorithm itself is invoked, depending on the mode of operation as described in Figure 3.11. As an acceptance strategy, only improving moves are admitted. If the position x^2 is chosen, then the previous velocity becomes useless. Hence, a new velocity is assigned to the particle using Equation (3.23).

$$v_{id}(t+1)= \alpha \; maxv \; r_5(t+1) \; , \tag{3.23}$$

Where, α is a constant number chosen as 0.01 or 0.001 in the experiments, *maxv* is the limit for the velocity of the particles, and $r_5(t)$ is a uniform random number in [-1,1] at time t.

If the fitness of a particle is not changed for a number of iterations; p, that is, variance of the fitness for the last p epoch is smaller than a threshold value, all of the particles within its neighborhood stop making random movements. Mode of operation switches to a refined search. In this mode of operation, particles start moving using the Equation (3.24) and Equation (3.25) when computing their second candidate positions.

$$v^2_{id}(t+1)=w \; v_{id}(t)+c_1 \; r_3(t) \; (y_{id}(t)-x_{id}(t))+c_2 \; r_4(t) \; (y_{gid}(t)-x_{id}(t)) \; , \tag{3.24}$$

$$x^2_{id}(t+1)=x_{id}(t)+v^2_{id}(t+1) \; , \tag{3.25}$$

4. EXPERIMENTS

4.1. Benchmark Functions

Well known benchmark functions that are used in the experiments are illustrated in Table 4.1. The names, sources, lower and upper bounds and number of global and local minima for these benchmark functions are provided in Table 4.2.

Table 4.1 Definition of the benchmark functions

Label	Equation
F1	$f(x) = 1-\sin(5\pi x)^6$
F2	$f(x) = 1-\exp(-2\log(2)((x-0.1)/0.8)^2)\sin(5\pi x)^6$
F3	$f(x)= 1-\sin(5\pi(x^{(3/4)}-0.05))^6$
F4	$f(x) = 1-(\exp(-2\log(2)*((x-0.08)/0.854)^2)\sin(5\pi\,(x^{(3/4)}-0.05))^6)$
F5	$f(x,y) = (x^2+y-11)^2-(x+y^2-7)^2$
F6	$f(x,y) = \sin(2.2\,\pi\,x+\pi/2)((\,2-\lvert y\rvert\,)/2)((\,3-\lvert x\rvert\,)/2) + \sin(0.5\pi y^2+\pi/2)((\,2-\lvert y\rvert\,)/2)((\,2-\lvert x\rvert\,)/2)$
F7	$f(x,y) = \cos(x)^2+\sin(y)^2$
F8	$f(x,y) = (\cos(2x+1)+2\cos(3x+2)+3\cos(4x+3)+4\cos(5x+4)+5\cos(6x+5))$ $(\cos(2y+1)+2\cos(3y+2)+3\cos(4y+3)+4\cos(5y+4)+5\cos(6y+5))$
F9	$f(x,y) = (y^2-4.5y^2)xy-4.7\cos(3x-y^2(2+x))\sin(2.5\pi x)+(0.3x)^2$
F10	$f(x,y) = 4x^2-2.1x^4+(1/3)x^6+xy-4y^2+4y^4$
F11	$f(\vec{x}) = \sum_{i=1}^{n-1} 100\cdot(x_{i+1}-x_i^2)^2+(x_i-1)^2$
F12	$f(\vec{x}) = \dfrac{1}{0.002+\sum_{j=1}^{25}\dfrac{1}{j+\sum_{i=1}^{2}(x_i-a_{ij})^6}}$ $a_{1j} = \begin{cases} -32 \to \mathrm{mod}(j,25)=1 \\ -16 \to \mathrm{mod}(j,25)=2 \\ 0 \to \mathrm{mod}(j,25)=3 \\ 16 \to \mathrm{mod}(j,25)=4 \\ 32 \to \mathrm{mod}(j,25)=0 \end{cases}$ $a_{2j} = \begin{cases} -32 \to j>0 \wedge j\le 5 \\ -16 \to j>5 \wedge j\le 10 \\ 0 \to j>10 \wedge j\le 15 \\ 16 \to j>15 \wedge j\le 20 \\ 32 \to j>20 \wedge j\le 5 \end{cases}$

F13	$f(\vec{x}) = 10 \cdot n + \sum_{i=1}^{n} (x_i^2 - 10 \cdot \cos(2\pi x_i))$
F14	$f(\vec{x}) = 6 \cdot n + \sum_{i=1}^{n} \lfloor x_i \rfloor$
F15	$f(\vec{x}) = 418.9829 \cdot n + \sum_{i=1}^{n} x_i \cdot \sin(\sqrt{\lvert x_i \rvert})$
F16	$f(\vec{x}) = \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod \cos(\frac{x_i}{\sqrt{i}}) + 1$
F17	$f(\vec{x}) = 20 + e - 20 \cdot e^{-0.2 \cdot \sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}} - e^{\frac{1}{n}\sum_{i=1}^{n} \cos(2\pi x_i)}$
F18	$f(\vec{x}) = -(\prod_{i=1}^{n} \cos(x_i)) \cdot (e^{-\sum_{i=1}^{n} (x_i - \pi)^2})$

Table 4.2. Benchmark functions used during the experiments; *minx* and *maxx* indicate the lower and upper bound, respectively; *dim* indicates the problem dimension.

Label	**Function Name**	***minx***	***maxx***	**dim**	**global**	**locals**	**Source**
F1	Deb's F1	0.00	1.00	1	5	0	[9][7][35][44]
F2	F2	0.00	1.00	1	1	4	[35][34]
F3	F3	0.00	1.00	1	5	0	[9][35][34]
F4	F4	0.00	1.00	1	1	4	[9][35][34]
F5	Himmelblau's F5	-10.00	10.00	2	4	0	[35][34][4]
F6	Ursem F3	-2.00	2.00	2	4	8	[45][4][46]
F7	F7	-4.00	4.00	2	6	0	[17]
F8	Shubert	-2.50	2.50	2	2	38	[7][47][4]
F9	M3-Waves	-1.2	1.2	2	1	9	[45]
F10	Six-Hump Camel	-1.9	1.9	2	2	4	[45][4]
F11	Rosenbrock	-2.048	2.048	10	1	0	[18]
F12	Foxhole	-65	65	2	1	many	[18]
F13	Rastrigin	-5.12	5.12	10	1	many	[18]
F14	Step	-5.12	5.12	10	1*	0	[18]
F15	Schwefel	-500	500	10	1	many	[48]
F16	Griewangk	-600	600	10	1	many	[49]

F17	Ackley	-32	32	10	1	many	[50]
F18	Easom	-100	100	10	1	0	[50]

For the dimensions and bounds given in Table 4.2, the benchmark functions have the following properties:

- Function F1 has 5 equal fitness global minima and they are evenly distributed along *x* axis (Figure 4.1(a)). The minimum points are at (0.1), (0.3), (0.5), (0.7), (0.9) with fitness value 0.
- Function F2 has 5 minima and they are evenly distributed along *x* axis. One of them is global minima and the others are local (Figure 4.1(b)). The minimum points are at (0.1), (0.3), (0.5), (0.7), (0.9) and the global optimum at 0.1 has fitness value 0.
- Function F3 has 5 equal fitness global minima and they are unevenly distributed along *x* axis (Figure 4.1(c)). The 5 global minima are at approximately (0.0797), (0.2465), (0.4506), (0.6814) and (0.9339) with fitness value 0.
- Function F4 has 5 minima and they are unevenly distributed along *x* axis. One of them is global minima and the others are local (Figure 4.1(d)). The global minima is at approximately (0.0797) with fitness value 0. The other minima, which are local, are at around (0.2463), (0.4495), (0.6792) and (0.9302).
- Function F5 has 4 equal fitness global minima with fitness values 0 (Figure 4.2). These optima are at approximately (-3.7793,-3.283186), (3.0,2.0), (2.8051,3.1313) and (3.5844,-1.8481).
- Function F6 has 4 equal fitness global minima with fitness values 0 and 8 local minima (Figure 4.3). The global minima are at (0.4411,-1.2047), (0.4411,1.2047), (-0.4411,1.2049) and (-0.4417,-1.2049); the local minima are at (1.3636,0.0), (-1.3636 0.0), (0.4546,0.0), (-0.4546,0.0), (1.3463,-1.114), (1.3463,1.114), (-1.3463,-1.114) and (-1.3463,1.114).
- Function F7 has 6 equal fitness global minima with fitness values 0 (Figure 4.4). They are approximately at (-1.5708,3.1416) (-1.5708,-3.1416), (1.5708,3.1416) (-1.5708,0.0), (1.5708,-3.1416) and (1.5708, 0.0).

- Function F8 has 2 global and 38 local minima (Figure 4.5). The global minima have fitness values 0 at (-0.8003,-1.4251) and (-1.4251,-0.8003).
- Function F9 has 1 global and 9 local minima (Figure 4.6). The global minimum is at (1.0296, -1.0787).
- Function F10 has 2 global and 4 local minima (Figure 4.7). The global minima are at (-0.089842,0.712656) and (0.089842,-0.712656).
- Function F11 has 1 global minima at (1,1) (Figure 4.8).
- Function F12 has 1 global minima around (-32,-32). It has 25 holes, and all have similar shape and fitness (Figure 4.9).
- Function F13 has 1 global minima at (0,0). But has several local minima (Figure 4.10).
- Function F14 has 1 global minima area around (-5,-5). It has a shape like a stairs (Figure 4.11).

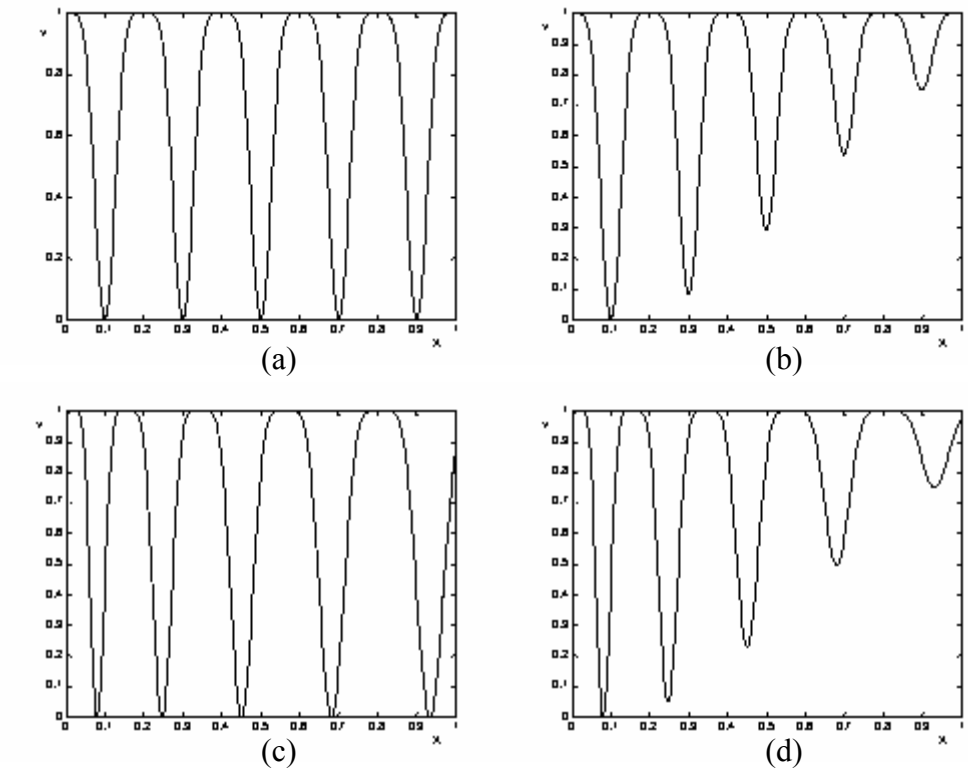


Figure 4.1. 2-D plot of functions (a) F1, (b) F2, (c) F3, (d) F4

- Function F15 has 1 global minima at (-421,-421), and several locals (Figure 4.12).
- Function F16 has 1 global minima at (0,0) and thousands of locals (Figure 4.13).
- Function F17 has 1 global minima at (0,0) and thousands of locals (Figure 4.14).
- Function F18 has 1 global minima near (3.14,3.14). It seems like a small hole in a flat surface (Figure 4.15).

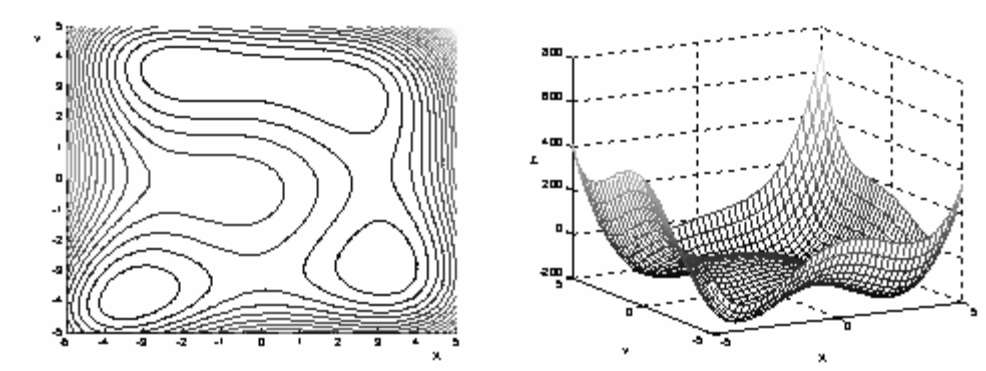


Figure 4.2. 3-D plot of function F5

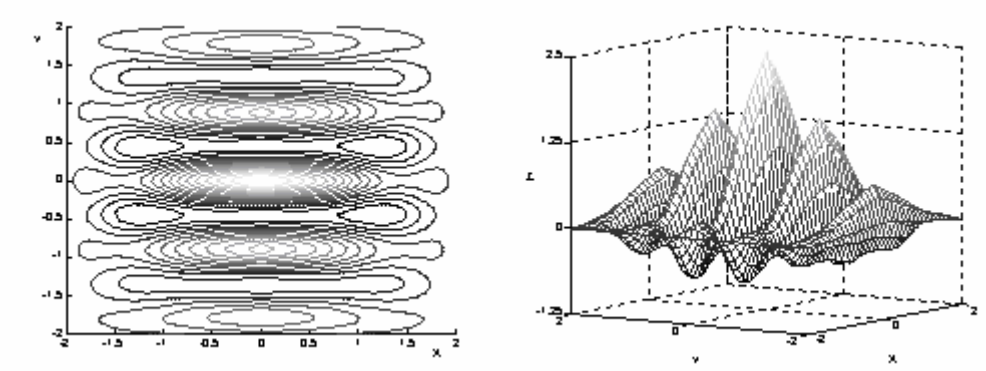


Figure 4.3. 3-D plot of function F6

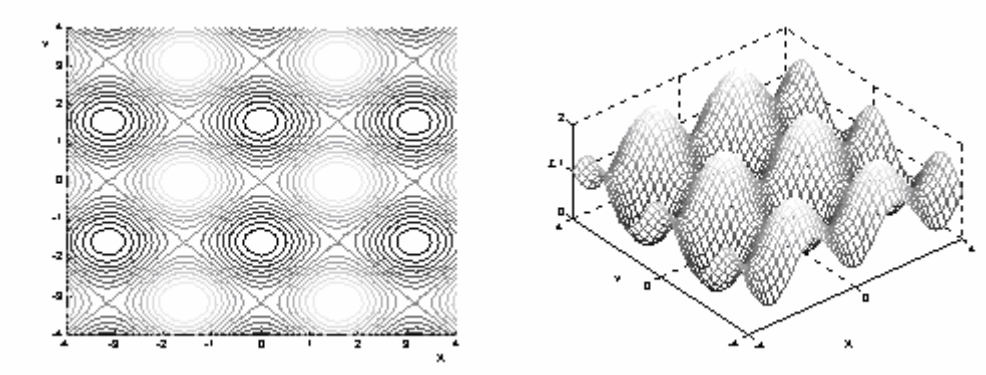


Figure 4.4. 3-D plot of function F7

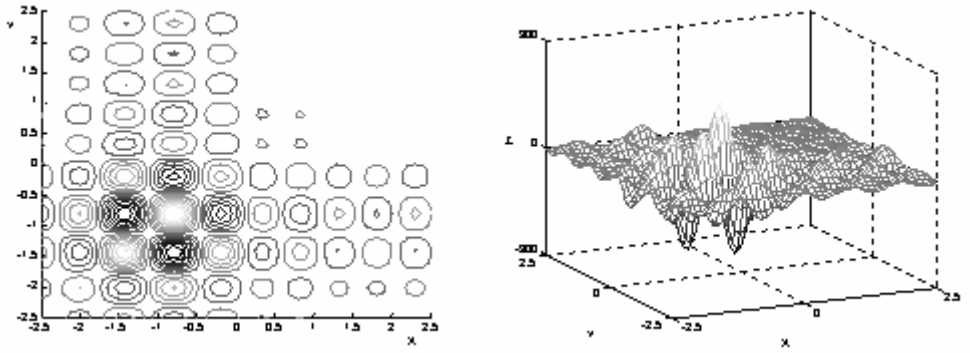


Figure 4.5. 3-D plot of function F8

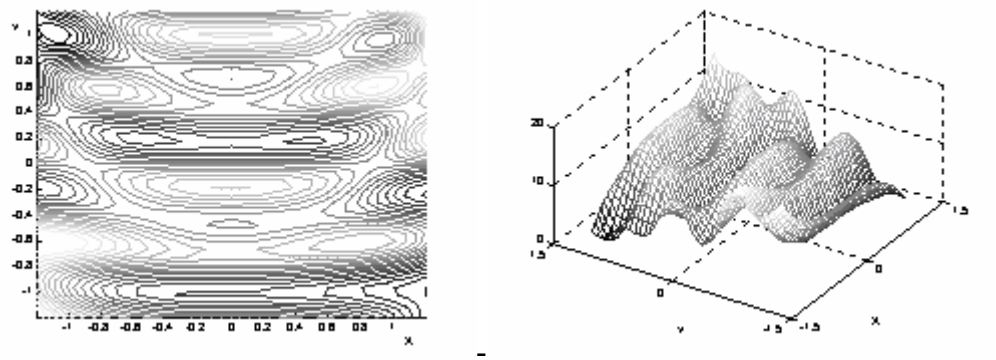


Figure 4.6. 3-D plot of function F9

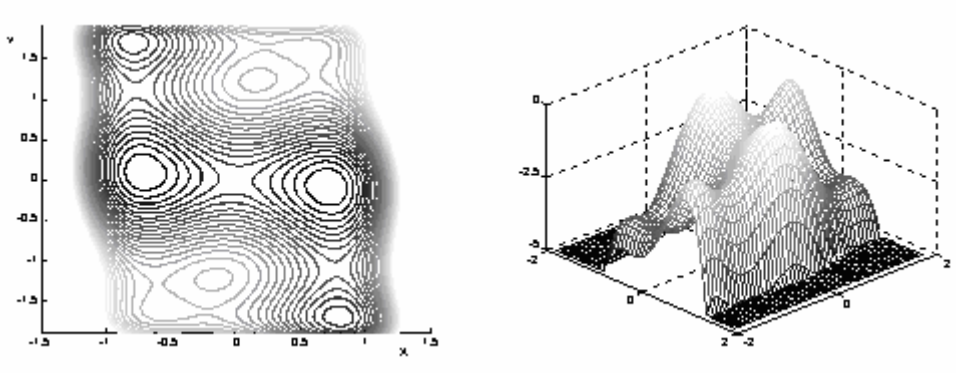


Figure 4.7. 3-D plot of function F10

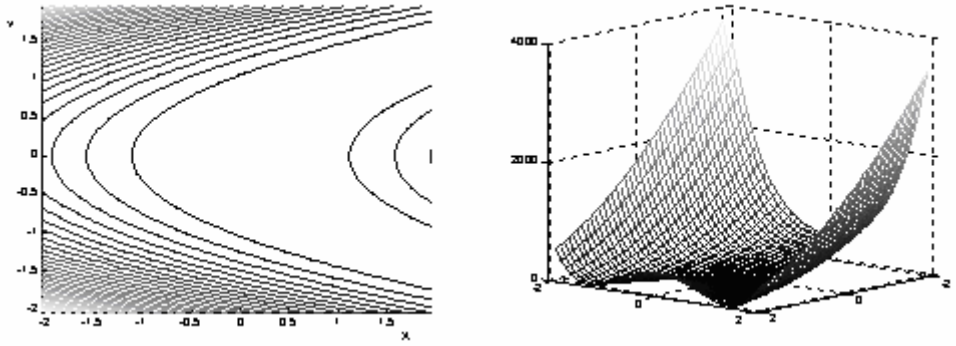


Figure 4.8. 3-D plot of function F11

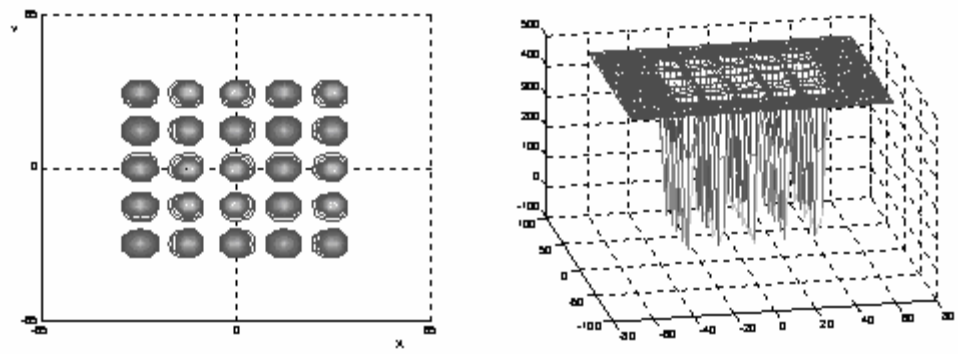


Figure 4.9. 3-D plot of function F12

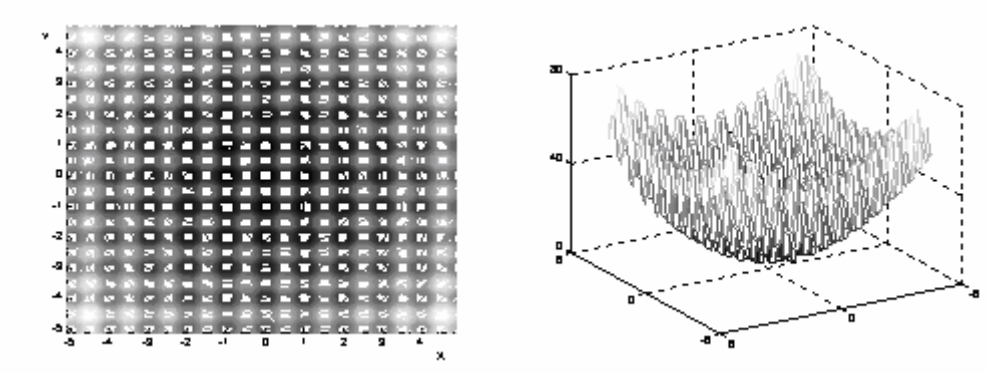


Figure 4.10. 3-D plot of function F13

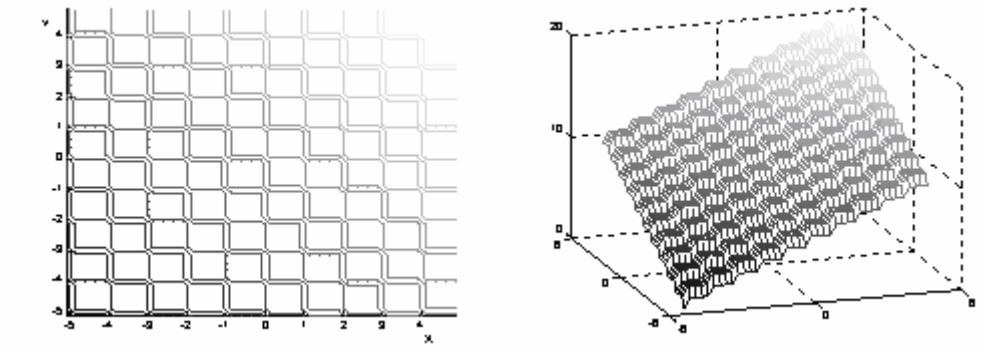


Figure 4.11. 3-D plot of function F14

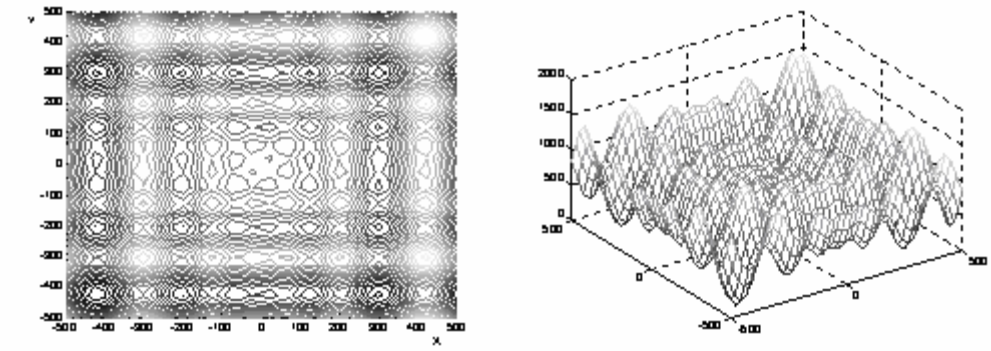


Figure 4.12. 3-D plot of function F15

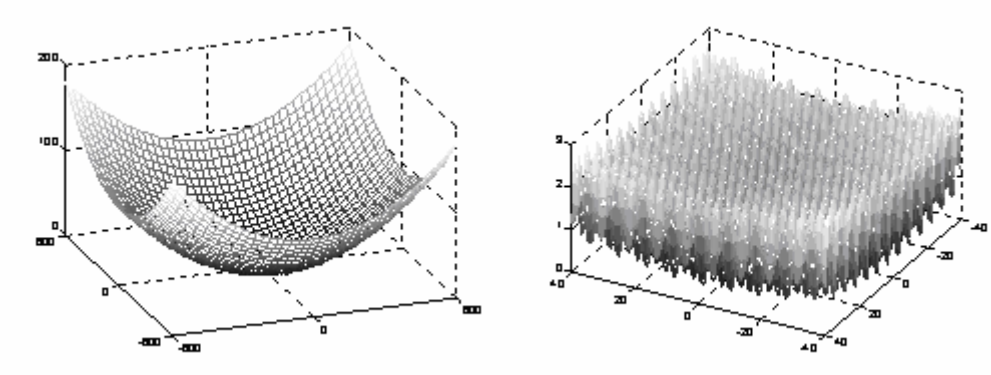


Figure 4.13. 3-D plot of function F16

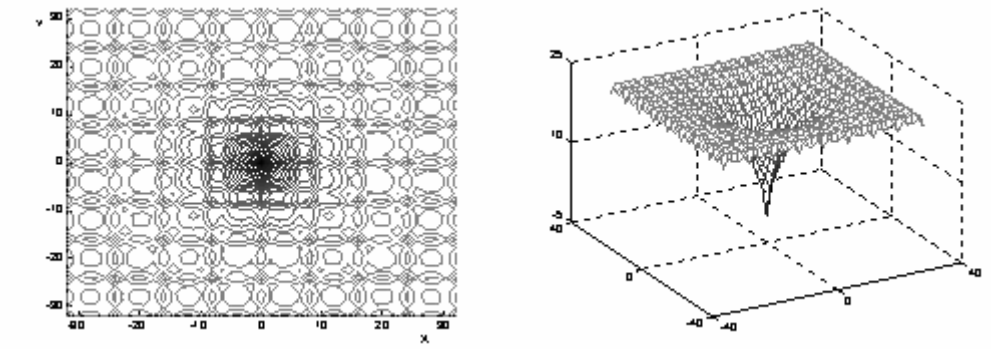


Figure 4.14 3-D plot of function F17

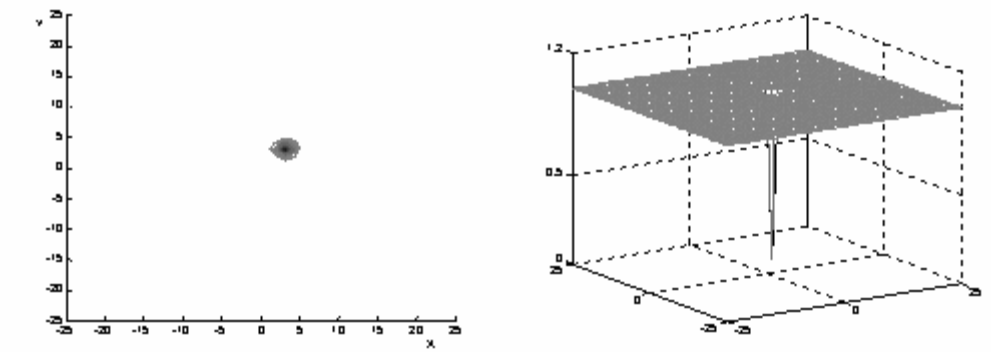


Figure 4.15. 3-D plot of function F18

The first ten functions are generally used on multimodal benchmark tests. The latter eight functions are generally used as a benchmark on the tests that aim to find one single global optimum.

4.2. Evaluation Criteria

In multimodal optimization problems, while having more than one optimum solution having equal quality, a predetermined subset of local optima might be required as well. Therefore, it is difficult to evaluate and compare different algorithms. Different evaluation algorithms are used by the scientists. 3 mostly used evaluation measurement criteria are:

1. *Number of evaluations* is the average number of function evaluations to find all of the global optimums.
2. *Peak ratio* is the ratio between the number of optima found and the total number of optimums. Peak ratio provides a criterion for measuring the capability of discovering (global or local) peaks.
3. *Peak accuracy* is calculated as: For each optimum, find the nearest particle *i* and add the absolute difference in fitness values between the optimum point and the particle *i* to the peak accuracy.

$$Peak\ accuracy = \sum_{j=1}^{\#optima} \left| \text{fitness}(peak_j) - \text{fitness}(i) \right| \tag{4.1}$$

Consistency is also an important feature of an algorithm. If several runs are performed, it is expected that the algorithm should locate optima as frequently as possible. *Success consistency ratio* denotes the proportion of successful runs to the total number of runs in which all global (*gscr*) and local optima (*lscr*) are discovered (Equations (4.4) and (4.5)).

In this thesis, peak ratios and success consistency ratios are used in order to evaluate the performance of the PSO algorithms. *Global peak ratio* (*gpr*) is defined as the proportion of the average number of global optima found to the total number of global optima (Equation (4.2)). Similarly, *local peak ratio* (*lpr*) is defined as presented in Equation (4.3).

$$gpr = \text{average \# of global optima found / \# of global optima} \tag{4.2}$$

$$lpr = \text{average \# of local optima found / \# of local optima} \quad (4.3)$$

$$gscr = \text{average \# of runs all global optima found / \# of runs} \quad (4.4)$$

$$lscr = \text{average \# of runs all local optima found / \# of runs} \quad (4.5)$$

Using the peak ratios and success consistency ratios, the *overall success ratio* (*osr*) is calculated as shown in Eq. 4.6. When there are no local optima, then, *lpr* and *lscr* are set to zero. Notice that *osr* will have values in the range [0,1]. A higher ratio indicates a better performance.

$$osr = \frac{(gpr + gscr)}{2} + \frac{(lpr + lscr)}{2} \quad (4.6)$$

4.3. Experimental Setup

All runs are performed on a 2 GHz, Windows 2003 operating system with 512 MB of memory. A Matlab code is generated for the experiments. Each experiment is repeated 50 times. The results given in this thesis are the average of these 50 runs of each configuration. A run is terminated either maximum number of evaluations is exceeded or all required global optima are found [17][7]. For a fair comparison, the maximum number of evaluations is limited during each test. The details of this limit are given in the following sections. In all algorithms except Unified PSO, inertia weight is linearly decreased from 0.8 down to 0.6 and constants c_1 and c_2 are set to 1.5 for a stable PSO. In Unified PSO, constriction factor X is used as 0.729, and c_1 and c_2 are set to 2.05. *CUTOFF* is set to 0.00005 for all functions.

To see the effect of the maximum velocity, on these tests, each algorithm is investigated with two different maximum velocities: "(*maxx-minx*)/20" and "(*maxx-minx*)/2", where each algorithm is labeled with respect to its maximum velocity component as the *algorithm_abbriviation*-20 and *algorithm_abbriviation*-2, respectively. Three main categories of tests are done using these parameters. The configurations of these tests are given in the following subsections.

4.3.1. Setup for Determining the Best Parameter Set for CPSO

In CPSO algorithm, 6 parameters are required. Intuitively, subswarm size should be as small as possible, just like the number of epochs, variance and Experiments are performed in order to determine the sensitivity of CPSO to these parameters with different values and to obtain the best set:

- The subswarm size $n \in \{2, 3, 4\}$
- The number of epochs $k \in \{3, 5, 7\}$ used during the craziness variance calculation
- The number of epochs $p \in \{3, 5, 7\}$ used during velocity reset variance calculation
- The $variance \in \{0.0001, 0.01\}$
- The number of epochs $m \in \{5,7,10\}$ for the subswarm reconstruction
- $\alpha \in \{0.001, 0.01\}$ velocity reset constant

Therefore, totally 648 different combinations of all parameters are tested, considering the maximum velocities as well. These tests are performed on F1-F10. Swarm size is chosen as 30 and 50, for functions F1-F4 and F5-F10, respectively. The maximum number of evaluations is limited to 15000 for F1-F4, and 25000 for F5-F10.

4.3.2. Setup for Multimodal Optimization Experiments

Seven different algorithms (SpecPSO, NichePSO, nbestPSO, CPSO, UniPSO, mNichePSO, PVPSO) are tested on 10 different benchmark functions (F1-F10). Swarm size is chosen as 30 and 50, for functions F1-F4 and F5-F10, respectively. Problem dimension is one for functions F1-F4, two for functions F5-F10. The maximum number of evaluations is limited to 15000 on F1-F4, and 25000 on F5-F10. In SpecPSO, species radius is chosen as "(*maxx-minx*)/20". In NichePSO, the parameters μ and variance are set

to 0.001 and 0.0001, respectively [35]. Additionally, in mNichePSO, the parameter maximum niche radius is set to (*maxx-minx*)/10. In nbestPSO, the neighborhood size is linearly decreased from 6 to 2. In CPSO, according to the experiments that are given in section 4.4.1 n, k, p, *variance*, m and α set to 2, 3, 3, 0.01, 5 and 0.01 respectively.

4.3.3. Setup for Global Optimization Experiments

In order to identify whether the proposed algorithm CPSO is suitable for global optimization or not, experiments are performed using the test functions F11-F18. CPSO is compared to compared to the Standard PSO and Unified PSO. Furthermore, in order to observe the scalability of each algorithm, the benchmark functions are tested using different dimensions: 2, 5, 10, 15 and 20. The swarm sizes and the maximum number of function evaluations with respect to these dimensions are set to 50, 50, 75, 75, 100, and 100000, 200000, 600000, 1200000, 2500000, respectively.

4.4. Experimental Results

In Section 4.4.1, Section 4.4.2 and Section 4.4.3, the consize results of the experiments for CPSO, multimodal and global optimization problems are discussed respectively. More on all the results obtained during the experiments can be found in Appendix A.

4.4.1. Best Parameter Set for the CPSO

For functions F1-F10, (see Section 4.3.1) 648 parameter set combinations are tested. These tests shown that, one parameter set combination can be chosen as a generic parameter set for all kinds of functions.

Table 4.3. Top 10 parameter sets with respect to the average *osr*

variance	***n***	***k***	***p***	***m***	***α***	***maxv***	**avg. *osr***
0.01	2	3	3	5	0.01	(*maxx-minx*)/20	0.8056
0.01	2	3	5	7	0.01	(*maxx-minx*)/20	0.8031
0.01	2	3	3	7	0.01	(*maxx-minx*)/20	0.7988
0.01	2	3	7	5	0.01	(*maxx-minx*)/20	0.7954

0.01	2	3	5	5	0.01	(*maxx-minx*)/20	0.7940
0.01	2	3	5	5	0.001	(*maxx-minx*)/20	0.7906
0.01	2	3	7	10	0.01	(*maxx-minx*)/20	0.7898
0.01	2	3	3	10	0.01	(*maxx-minx*)/20	0.7886
0.01	2	3	7	7	0.01	(*maxx-minx*)/20	0.7882
0.01	2	3	3	5	0.001	(*maxx-minx*)/20	0.7879

Each parameter set is evaluated with respect to their *osr* (overall success ratio) for each function. Then, the average *osr* of each is computed. The 10 parameter sets which have the maximum *osr* are provided in Table 4.3. So, the parameter set {0.01, 2, 3, 3, 5, 0.01, (*maxx-minx*)/20} turns out to be the best choice for variance, n, k, p, m, α and the maximum velocity in CPSO, respectively. Further experiments are performed using this parameter set. It seems that small values for n, k, p, m and the maximum velocity and large values for the variance and α improves the performance slightly.

4.4.2. Results for Multimodal Optimization Problems

Two different maximum velocity values are used in these tests: (*maxx-minx*)/2 and (*maxx-minx*)/20. Therefore, to show which value for the maximum velocity is used, the divisor in the formulas is appended to the algorithm abbreviation. To examine the effect of the maximum velocity and to determine what kind of maximum velocity is suitable for each algorithm, the experimental results are provided together.

Table 4.4. Average *osr* of each algorithm over all runs for each benchmark function

Algorithm	**F1**	**F2**	**F3**	**F4**	**F5**	**F6**	**F7**	**F8**	**F9**	**F10**
Spec-2	0.976	0.823	0.964	0.838	**1.000**	0.638	0.965	0.579	0.648	0.593
Spec-20	0.988	0.994	**1.000**	0.941	0.950	0.478	0.715	0.269	0.636	0.823
Niche-2	0.452	0.688	0.396	0.639	0.125	0.095	0.147	0.130	0.593	0.140
Niche-20	0.974	**1.000**	0.940	0.961	0.190	0.214	0.445	0.133	0.660	0.451
nbest-2	0.710	0.854	0.770	0.799	0.885	0.471	0.723	0.411	0.519	0.595
nbest-20	0.808	0.933	0.744	0.843	0.898	0.426	0.737	0.312	0.461	0.575
CPSO-2	0.964	0.781	0.830	0.778	0.278	0.563	0.977	0.173	0.562	0.533
CPSO-20	0.964	0.873	0.904	0.900	**1.000**	0.656	**1.000**	0.517	0.627	0.615
Unified-2	0.716	0.695	0.658	0.695	0.813	0.474	0.643	0.507	0.564	0.503
Unified-20	0.720	0.805	0.678	0.744	0.875	0.541	0.552	0.479	0.449	0.539
mNiche-2	0.696	0.688	0.608	0.666	**1.000**	0.539	0.907	**0.619**	**0.749**	0.635
mNiche-20	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**0.660**	0.988	0.135	0.469	**0.909**
PVPSO-2	0.988	0.925	0.976	0.969	0.445	0.565	0.790	0.393	0.546	0.623
PVPSO-20	**1.000**	**1.000**	**1.000**	**1.000**	0.205	0.377	0.455	0.196	0.354	0.705

Tables 4.4 present the overall success rates of the algorithms on all 10 functions. Bold entries in the table point out the best algorithm(s) for the corresponding function. None of the algorithms is capable of finding all optima on all functions. Therefore, to see which algorithm has greater overall success rate, the averages are calculated and presented in Table 4.5. The proposed modification to NichePSO, mNichePSO, provides much better performance than the original considering the average *osr*. Furthermore, mNiche-20, CPSO-20 and Spec-2 are the top three algorithms in that order with respect to the average overall success ratio. PVPSO-20 is as successful as mNiche-20 for locating multiple optima in 2D search lanscapes, while its performance deteriorates extremely for 3D search landscapes. mNiche, CPSO and PVPSO are more sensitive to the maximum velocity as compared to the others.

Table 4.5. Overall average *osr* and its standard deviation over the problem instances for each algorithm

Algorithm	*osr*	
	avr.	**stdev.**
Spec-2	0.802	0.173
Spec-20	0.779	0.252
Niche-2	0.340	0.240
Niche-20	0.597	0.355
nbest-2	0.674	0.165
nbest-20	0.674	0.216
CPSO-2	0.644	0.272
CPSO-20	0.806	0.182
Unified-2	0.627	0.111
Unified-20	0.638	0.146
mNiche-2	0.711	0.142
mNiche-20	0.816	0.301
PVPSO-2	0.722	0.234
PVPSO-20	0.629	0.348

4.4.3. Results for Global Optimization Problems

The performance of the algorithms on global optimization problems, two main criteria are used: the (global) success rate and the average best fitness. The experimental results are presented in Fig. 4-16. The average best fitness change is demonsrated as the

dimensionality is increased. The average *osr* of each algorithm on these problems are also summarized in Table 4.6.

The results show that, Standard PSO and Unified PSO algorithms perform well and have a good scaling on global optimization problems, while the proposed algorithm CPSO seems not to be suitable for global optimization problems. The rest of the detailed experimental results are presented in Appendix A. Still, with some tuning CPSO might provide better results. Note that for low dimensions CPSO is promising (Table 4.6).

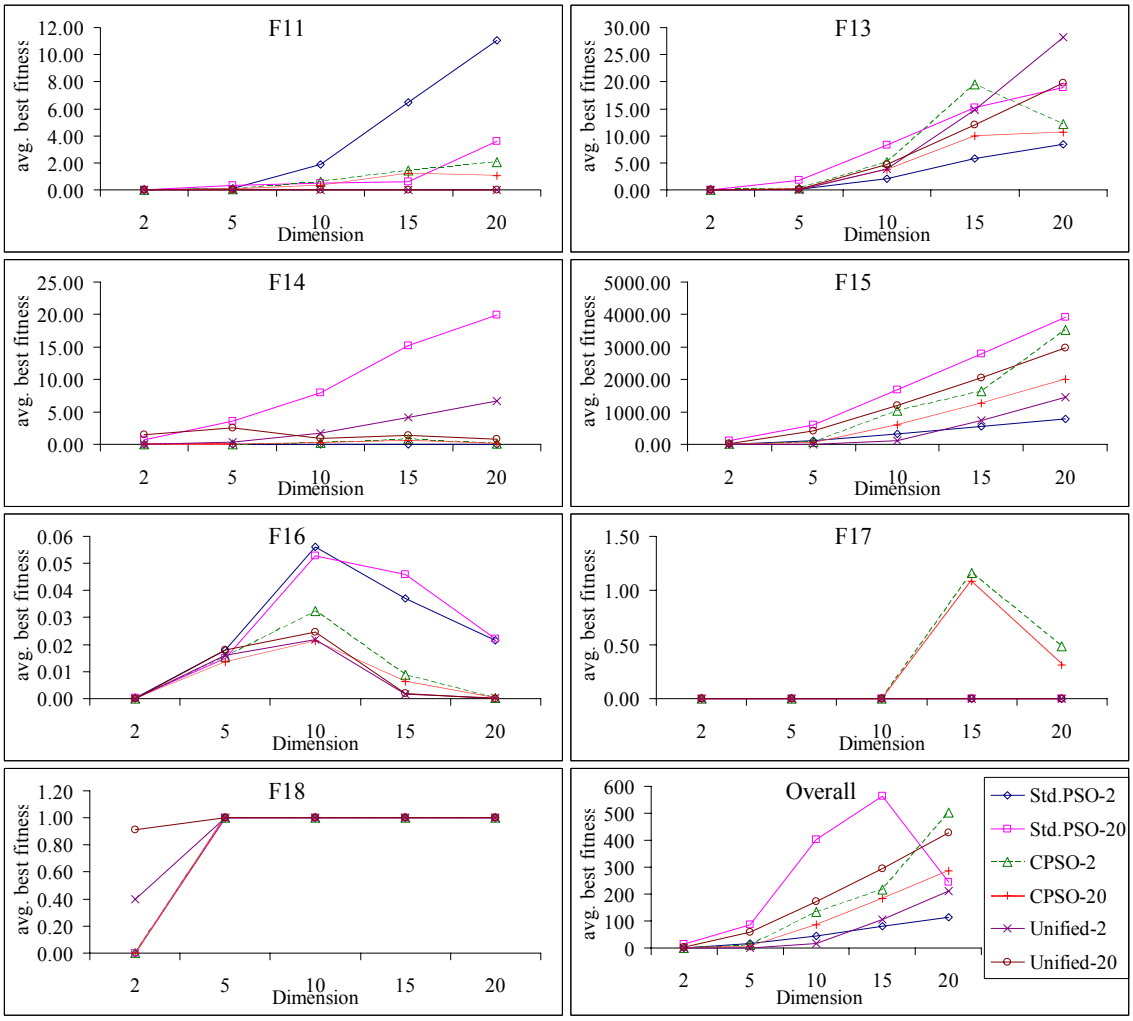


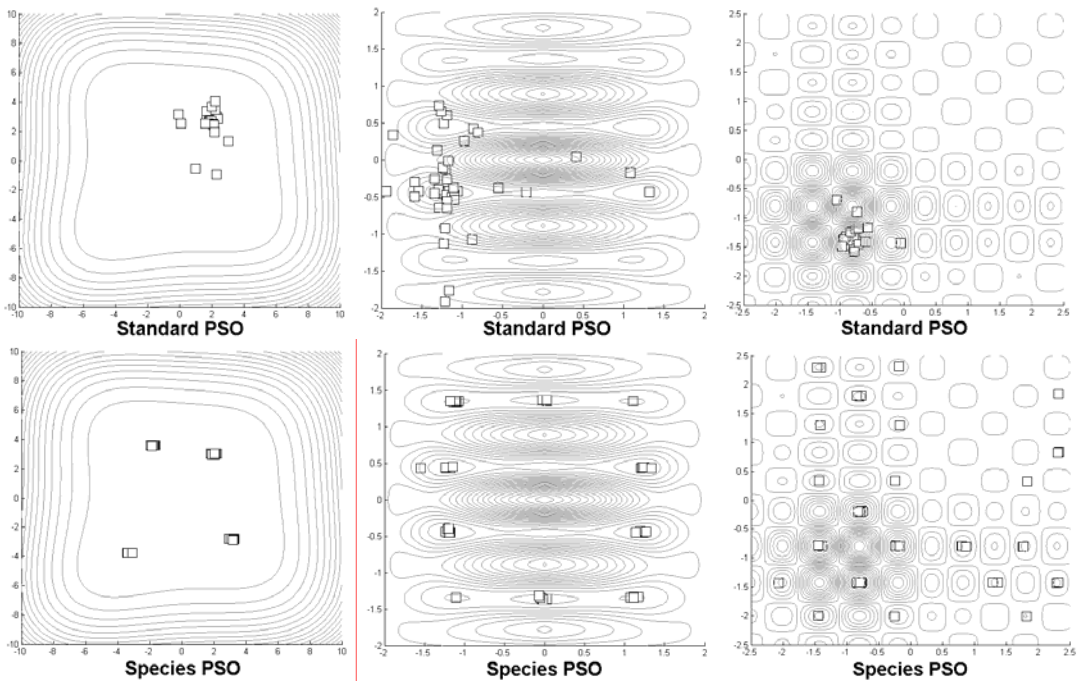
Figure 4.16. Plot of average best fitness versus dimension for each benchmark function for global optimization

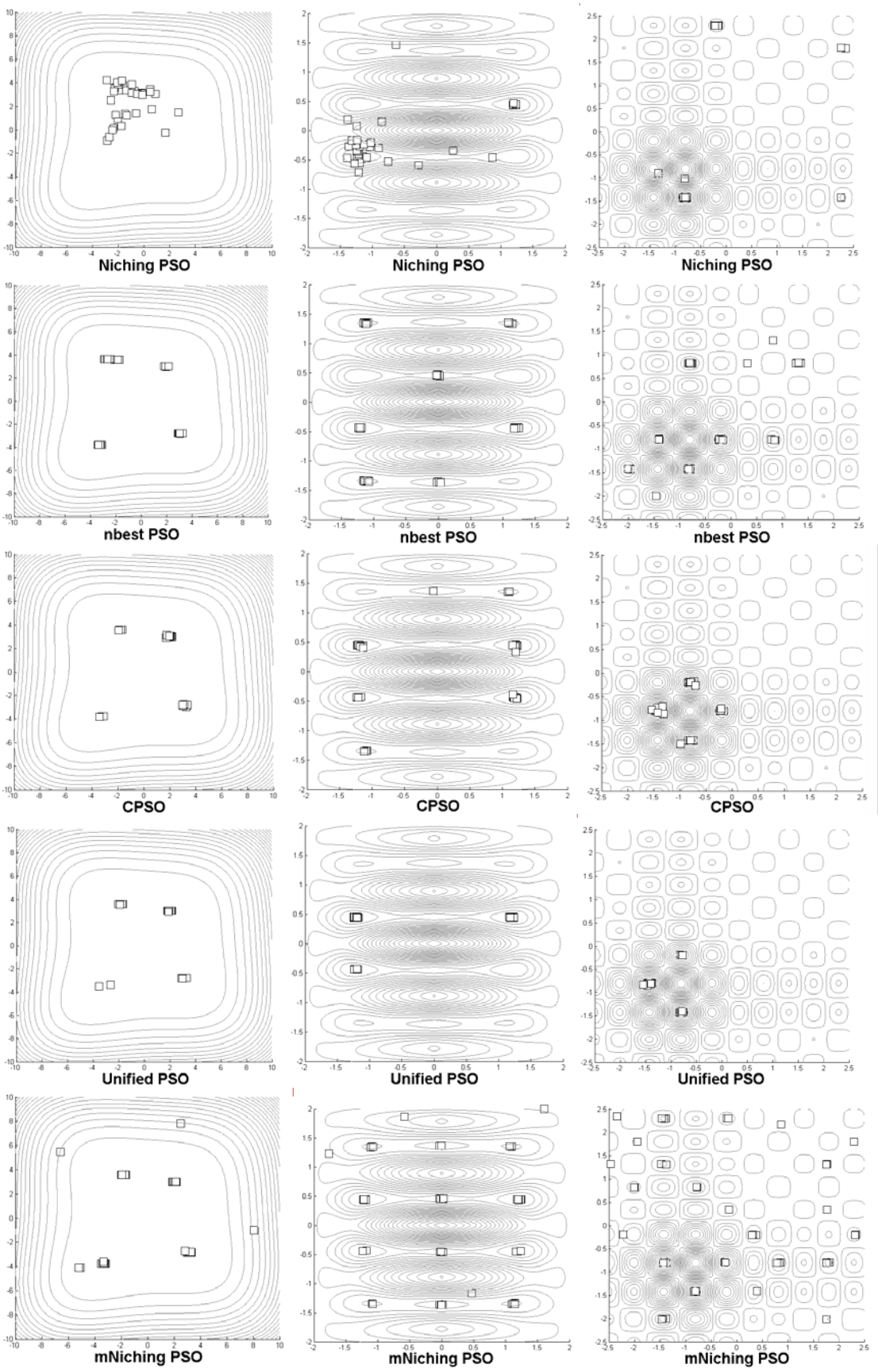
Table 4.6. Success rate of each algorithm for each benchmark function for 2-D

Algorithm	F11	F12	F13	F14	F15	F16	F17	F18	avr.
Std.PSO-2	**1.00**	**1.00**	**1.00**	**1.00**	0.96	0.94	**1.00**	**1.00**	**0.99**
Std.PSO-20	**1.00**	0.40	**1.00**	0.40	0.26	0.96	**1.00**	**1.00**	0.75
CPSO-2	**1.00**	0.98	**1.00**	**1.00**	**1.00**	**1.00**	0.88	**1.00**	0.98
CPSO-20	**1.00**	0.98	**1.00**	**1.00**	**1.00**	**1.00**	0.88	**1.00**	0.98
Unified-2	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	0.60	0.95
Unified-20	**1.00**	0.92	**1.00**	0.20	0.80	**1.00**	**1.00**	0.06	0.75

4.5. Discussions on the Experimental Results

Standard PSO algorithm is not capable of finding more than one optimum in parallel. An iterative approach should be utilized if more than one optimum is needed. If the problem is multimodal, standard PSO algorithm may get stuck at a local optimum which is not desired. As presented in Figure 4.17, the diversity of the particles cannot be maintained using a standard PSO algorithm.




Niching PSO
Niching PSO
Niching PSO
nbest PSO
nbest PSO
nbest PSO
CPSO
CPSO
CPSO
Unified PSO
Unified PSO
Unified PSO
mNiching PSO
mNiching PSO
mNiching PSO

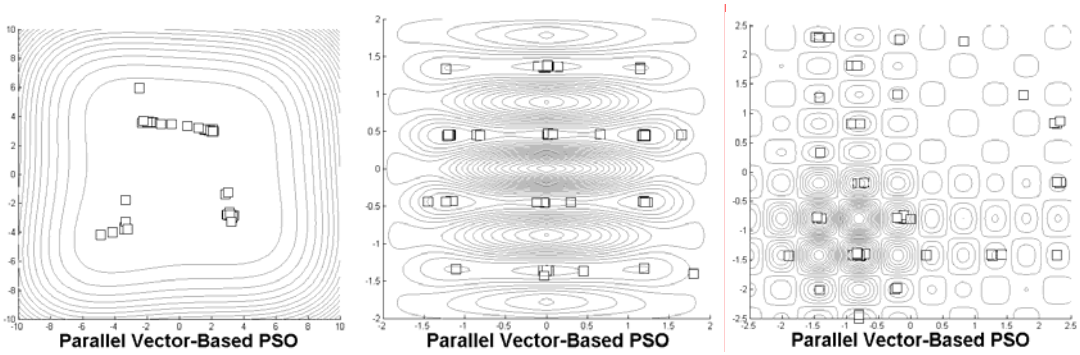
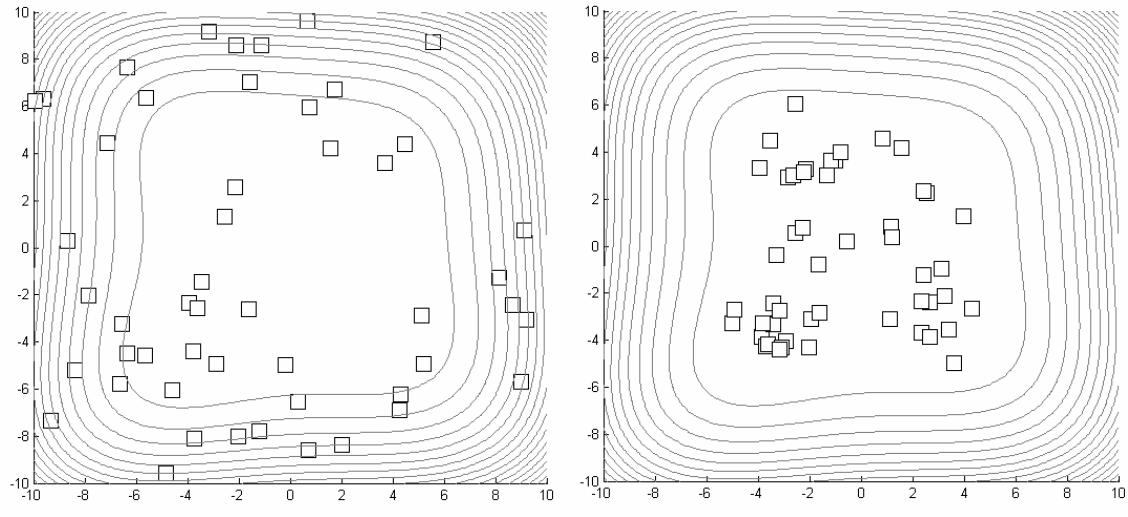


Figure 4.17. Sample runs of all algorithms on functions F5, F6 and F8 for 2-*D*

Species-based PSO is the best performing algorithm for multimodal optimization. It is good at locating multiple local and global optima in parallel. On most of the problems, the species were evenly distributed to the search space and founded the required optima.

The original Niching PSO sometimes provides good separation of niches. But sometimes, all of the niches are merged together. Consequently, they can only find a few of the optima. When a restriction is added to the algorithm to prevent the niche radius to grow, as mNichePSO, the merging of the niches will not occur as frequently. Therefore, a better diversity can be obtained.

Although, the *nbest* PSO is proposed for solving systems of unconstrained equations, it can solve some multimodal optimization problems. It provides a good diversity, but it has a very bad convergence rate. The proposed CPSO algorithm is very successful in locating global optima. The hill-climbing strategy gives good exploitation capability, and random-walk property provides a superior exploration capability. As in Figure 4.18, CPSO explore the whole search space and find the existing optima. The Unified PSO algorithm can be easily implemented, and can find multiple optima which may be local or global. Its parameters provide a simple way to force the algorithm to find multiple or single optimum. The Parallel Vector-Based PSO algorithm has the ability to find global and local optima, but may not be preferred due to its poor average overall success ratio.



(a) iterariton #1

(b) iterarion #5

(c) iteration #25

(d) iteration #37

Figure 4.18. A Sample run of CPSO on function F5

5. CONCLUSIONS

Previous studies on PSO algorithms based on subpopulations for multimodal optimization are reviewed. There are six essential PSO algorithms that have been proposed for multimodal optimization: Species-based PSO, Niching PSO, *nbest* PSO, Stretching PSO, Unified PSO and Parallel Vector-Based PSO. During the preliminary experiments, it has been observed that the performance of NichePSO can be improved significantly by a simple modification. The niche radius is restricted so that it can not exceed a predefined value. This algorithm is referred as mNichePSO. mNichePSO performs much better than NichePSO on the benchmark functions, considering the evaluation criteria. Furthermore, a new multimodal particle swarm optimization algorithm called CPSO is introduced. CPSO is compared against these approaches. With its enhanced exploration and exploitation capabilities, CPSO has a good performance on finding all global optima, like mNichePSO and SpecPSO.

The proposed heuristics *peak ratio*, *success consistency rate* and *overall success ratio* are used as the evaluation criteria which facilitated the comparison of the algorithms. All algorithms introduce additional parameters. It is hard to obtain an optimal set of parameters. Still, fine tuning these parameters can help to improve the performance for a given problem. All of the above algorithms, except the Unified PSO and the standard PSO, are computationally expensive, requiring distance calculation. If the dimensionality increases, the scalability issue might arise, causing this cost to become remarkable. In CPSO, this cost is somewhat reduced by making the additional computations at a predefined frequency.

Experiments shown that, overall performance of mNichePSO is better than other algorithms for multimodal problems. But, depending on the problem type, best performing algorithm changes most of the time. This observation confirms the No Free Lunch Theorem [52]. A future research on the implementation of a meta-heuristic, that chooses the appropriate algorithm among existing ones, may help to locate optima more efficiently.

APPENDIX A: DETAILS OF THE EXPERIMENTAL RESULTS

Table A.1. Results for each parameter set experimented in CPSO

var.	n	k	p	m	α	maxv	avg. gpr	avg. lpr	avg. gscr	avg. lscr	avg. osr
0.01	2	3	3	5	0.01	(maxx-minx)/20	0.9951	0.7498	0.9760	0.5280	0.8056
0.01	2	3	5	7	0.01	(maxx-minx)/20	0.9973	0.7240	0.9860	0.5220	0.8031
0.01	2	3	3	7	0.01	(maxx-minx)/20	0.9957	0.7280	0.9780	0.5200	0.7988
0.01	2	3	7	5	0.01	(maxx-minx)/20	0.9922	0.7407	0.9640	0.5260	0.7954
0.01	2	3	5	5	0.01	(maxx-minx)/20	0.9937	0.7407	0.9680	0.5120	0.7940
0.01	2	3	5	5	0.001	(maxx-minx)/20	0.9921	0.7362	0.9620	0.5180	0.7906
0.01	2	3	7	10	0.01	(maxx-minx)/20	0.9945	0.7081	0.9800	0.4980	0.7898
0.01	2	3	3	10	0.01	(maxx-minx)/20	0.9941	0.7123	0.9720	0.5100	0.7886
0.01	2	3	7	7	0.01	(maxx-minx)/20	0.9925	0.7259	0.9620	0.5180	0.7882
0.01	2	3	3	5	0.001	(maxx-minx)/20	0.9910	0.7376	0.9560	0.5200	0.7879
0.01	2	3	7	7	0.001	(maxx-minx)/20	0.9925	0.7276	0.9620	0.5140	0.7877
0.01	2	3	5	10	0.001	(maxx-minx)/20	0.9933	0.7070	0.9680	0.5140	0.7859
0.01	2	5	3	7	0.01	(maxx-minx)/20	0.9977	0.6894	0.9880	0.4740	0.7837
0.01	2	3	5	10	0.01	(maxx-minx)/20	0.9921	0.7134	0.9620	0.5080	0.7824
0.01	2	5	5	5	0.01	(maxx-minx)/20	0.9953	0.7047	0.9760	0.4760	0.7808
0.01	2	3	7	5	0.001	(maxx-minx)/20	0.9915	0.7295	0.9580	0.4920	0.7801
0.01	2	5	3	5	0.01	(maxx-minx)/20	0.9944	0.7011	0.9720	0.4860	0.7800
0.01	2	3	7	10	0.001	(maxx-minx)/20	0.9928	0.7065	0.9660	0.4940	0.7795
0.01	2	5	7	5	0.01	(maxx-minx)/20	0.9940	0.7039	0.9700	0.4860	0.7795
0.01	2	5	5	7	0.01	(maxx-minx)/20	0.9955	0.6902	0.9780	0.4760	0.7783
0.0001	2	3	7	5	0.01	(maxx-minx)/20	0.9961	0.6680	0.9800	0.4820	0.7755
0.01	2	3	3	10	0.001	(maxx-minx)/20	0.9933	0.6948	0.9660	0.4880	0.7754
0.01	2	3	5	7	0.001	(maxx-minx)/20	0.9898	0.7136	0.9580	0.4840	0.7749
0.0001	2	3	3	5	0.01	(maxx-minx)/20	0.9950	0.6751	0.9740	0.4860	0.7748
0.01	2	5	5	5	0.001	(maxx-minx)/20	0.9935	0.6981	0.9700	0.4700	0.7744
0.01	2	5	7	7	0.01	(maxx-minx)/20	0.9940	0.6925	0.9700	0.4740	0.7743
0.01	2	7	3	7	0.01	(maxx-minx)/20	0.9964	0.6717	0.9820	0.4660	0.7736
0.01	2	7	5	7	0.01	(maxx-minx)/20	0.9960	0.6669	0.9800	0.4660	0.7712
0.01	2	5	3	5	0.001	(maxx-minx)/20	0.9940	0.6931	0.9720	0.4580	0.7708
0.01	2	3	3	7	0.001	(maxx-minx)/20	0.9891	0.7140	0.9460	0.4960	0.7700
0.01	2	5	5	10	0.01	(maxx-minx)/20	0.9945	0.6790	0.9720	0.4680	0.7700
0.0001	2	3	5	5	0.01	(maxx-minx)/20	0.9929	0.6742	0.9640	0.4880	0.7690
0.0001	2	5	5	5	0.01	(maxx-minx)/20	0.9964	0.6517	0.9820	0.4660	0.7686
0.01	2	5	3	10	0.01	(maxx-minx)/20	0.9947	0.6760	0.9740	0.4600	0.7684
0.01	2	5	7	5	0.001	(maxx-minx)/20	0.9928	0.6926	0.9640	0.4660	0.7681
0.0001	2	3	7	7	0.01	(maxx-minx)/20	0.9956	0.6622	0.9780	0.4600	0.7674
0.0001	2	3	3	10	0.01	(maxx-minx)/20	0.9953	0.6569	0.9760	0.4680	0.7669
0.01	2	7	7	7	0.001	(maxx-minx)/20	0.9960	0.6594	0.9800	0.4560	0.7669
0.01	2	5	7	10	0.01	(maxx-minx)/20	0.9930	0.6821	0.9640	0.4700	0.7665
0.01	2	7	3	5	0.001	(maxx-minx)/20	0.9948	0.6715	0.9760	0.4520	0.7663
0.0001	2	3	5	5	0.001	(maxx-minx)/20	0.9941	0.6643	0.9700	0.4700	0.7656
0.01	2	7	5	5	0.01	(maxx-minx)/20	0.9941	0.6762	0.9700	0.4560	0.7651

0.0001	2	3	5	7	0.01	(maxx-minx)/20	0.9948	0.6609	0.9740	0.4620	0.7651
0.01	2	7	3	5	0.01	(maxx-minx)/20	0.9933	0.6790	0.9660	0.4620	0.7649
0.01	3	3	3	5	0.01	(maxx-minx)/20	0.9887	0.6954	0.9460	0.4920	0.7648
0.0001	2	3	3	5	0.001	(maxx-minx)/20	0.9941	0.6647	0.9700	0.4660	0.7647
0.01	2	7	7	5	0.001	(maxx-minx)/20	0.9945	0.6634	0.9720	0.4600	0.7641
0.01	2	5	7	7	0.001	(maxx-minx)/20	0.9935	0.6763	0.9680	0.4540	0.7640
0.01	2	7	7	10	0.01	(maxx-minx)/20	0.9939	0.6580	0.9720	0.4660	0.7639
0.01	2	7	7	5	0.01	(maxx-minx)/20	0.9937	0.6761	0.9680	0.4560	0.7639
0.01	2	7	5	5	0.001	(maxx-minx)/20	0.9952	0.6582	0.9760	0.4540	0.7637
0.01	2	5	7	10	0.001	(maxx-minx)/20	0.9944	0.6649	0.9740	0.4520	0.7634
0.01	2	7	5	7	0.001	(maxx-minx)/20	0.9953	0.6589	0.9760	0.4520	0.7634
0.0001	2	5	5	10	0.01	(maxx-minx)/20	0.9968	0.6315	0.9840	0.4560	0.7623
0.01	2	7	5	10	0.01	(maxx-minx)/20	0.9960	0.6509	0.9800	0.4460	0.7622
0.0001	2	5	7	7	0.01	(maxx-minx)/20	0.9956	0.6373	0.9800	0.4600	0.7621
0.0001	2	3	3	7	0.01	(maxx-minx)/20	0.9956	0.6499	0.9780	0.4500	0.7618
0.0001	2	7	7	5	0.01	(maxx-minx)/20	0.9961	0.6396	0.9800	0.4520	0.7609
0.0001	2	3	7	7	0.001	(maxx-minx)/20	0.9951	0.6485	0.9760	0.4520	0.7607
0.01	2	5	5	7	0.001	(maxx-minx)/20	0.9920	0.6760	0.9620	0.4580	0.7605
0.01	3	3	5	5	0.01	(maxx-minx)/20	0.9868	0.6867	0.9360	0.5040	0.7603
0.0001	2	7	3	7	0.01	(maxx-minx)/20	0.9957	0.6370	0.9800	0.4520	0.7601
0.01	3	3	3	10	0.01	(maxx-minx)/20	0.9885	0.6693	0.9460	0.5020	0.7601
0.0001	2	7	5	5	0.01	(maxx-minx)/20	0.9949	0.6393	0.9740	0.4600	0.7592
0.01	2	5	3	10	0.001	(maxx-minx)/20	0.9934	0.6595	0.9660	0.4580	0.7591
0.0001	2	5	7	5	0.01	(maxx-minx)/20	0.9945	0.6451	0.9720	0.4580	0.7590
0.01	3	3	7	5	0.001	(maxx-minx)/20	0.9877	0.6865	0.9460	0.4820	0.7589
0.0001	2	3	7	10	0.01	(maxx-minx)/20	0.9945	0.6480	0.9720	0.4520	0.7582
0.0001	2	7	5	5	0.001	(maxx-minx)/20	0.9956	0.6357	0.9800	0.4460	0.7582
0.0001	2	5	3	10	0.01	(maxx-minx)/20	0.9972	0.6264	0.9860	0.4380	0.7577
0.01	2	7	3	10	0.01	(maxx-minx)/20	0.9927	0.6547	0.9640	0.4620	0.7575
0.01	2	3	3	10	0.001	(maxx-minx)/2	0.9951	0.6177	0.9760	0.4700	0.7574
0.01	2	7	7	7	0.01	(maxx-minx)/20	0.9923	0.6650	0.9620	0.4540	0.7569
0.01	2	3	5	7	0.001	(maxx-minx)/2	0.9962	0.6107	0.9820	0.4560	0.7564
0.01	2	5	3	7	0.001	(maxx-minx)/20	0.9905	0.6682	0.9560	0.4640	0.7563
0.0001	2	7	7	7	0.001	(maxx-minx)/20	0.9964	0.6217	0.9840	0.4420	0.7561
0.0001	2	3	3	7	0.001	(maxx-minx)/20	0.9941	0.6484	0.9700	0.4480	0.7561
0.0001	2	7	5	7	0.001	(maxx-minx)/20	0.9965	0.6211	0.9820	0.4460	0.7560
0.0001	2	5	3	5	0.01	(maxx-minx)/20	0.9937	0.6484	0.9700	0.4480	0.7559
0.0001	2	7	3	5	0.01	(maxx-minx)/20	0.9953	0.6324	0.9780	0.4440	0.7557
0.0001	2	5	5	7	0.01	(maxx-minx)/20	0.9945	0.6372	0.9720	0.4520	0.7555
0.01	2	7	7	10	0.001	(maxx-minx)/20	0.9940	0.6418	0.9700	0.4520	0.7555
0.01	3	3	3	7	0.01	(maxx-minx)/20	0.9858	0.6927	0.9300	0.4960	0.7550
0.0001	2	5	7	7	0.001	(maxx-minx)/20	0.9952	0.6294	0.9760	0.4480	0.7549
0.0001	2	7	7	7	0.01	(maxx-minx)/20	0.9948	0.6269	0.9740	0.4540	0.7546
0.0001	2	5	7	5	0.001	(maxx-minx)/20	0.9929	0.6402	0.9680	0.4560	0.7545
0.01	2	3	5	5	0.001	(maxx-minx)/2	0.9949	0.6155	0.9740	0.4640	0.7543
0.0001	2	5	3	7	0.01	(maxx-minx)/20	0.9945	0.6378	0.9720	0.4460	0.7542
0.0001	2	7	5	7	0.01	(maxx-minx)/20	0.9960	0.6229	0.9800	0.4420	0.7542
0.0001	2	7	3	10	0.01	(maxx-minx)/20	0.9957	0.6189	0.9780	0.4500	0.7540
0.0001	2	7	7	10	0.01	(maxx-minx)/20	0.9952	0.6177	0.9780	0.4500	0.7535
0.01	2	3	3	5	0.001	(maxx-minx)/2	0.9957	0.6165	0.9780	0.4500	0.7535
0.0001	2	7	7	5	0.001	(maxx-minx)/20	0.9945	0.6299	0.9720	0.4500	0.7532

0.01	2	5	5	10	0.001	(maxx-minx)/20	0.9925	0.6573	0.9620	0.4460	0.7530
0.01	3	5	5	5	0.01	(maxx-minx)/20	0.9873	0.6756	0.9360	0.4880	0.7525
0.01	2	7	3	7	0.001	(maxx-minx)/20	0.9904	0.6613	0.9540	0.4600	0.7525
0.01	2	7	3	10	0.001	(maxx-minx)/20	0.9956	0.6373	0.9780	0.4240	0.7521
0.0001	2	3	5	10	0.01	(maxx-minx)/20	0.9936	0.6435	0.9680	0.4360	0.7507
0.01	3	3	5	7	0.01	(maxx-minx)/20	0.9865	0.6713	0.9340	0.4900	0.7506
0.0001	2	7	3	7	0.001	(maxx-minx)/20	0.9952	0.6237	0.9760	0.4360	0.7505
0.0001	2	5	3	5	0.001	(maxx-minx)/20	0.9941	0.6391	0.9700	0.4340	0.7503
0.0001	2	3	7	10	0.001	(maxx-minx)/20	0.9921	0.6422	0.9620	0.4500	0.7501
0.01	2	3	5	10	0.001	(maxx-minx)/2	0.9941	0.6060	0.9700	0.4660	0.7501
0.01	3	5	5	5	0.001	(maxx-minx)/20	0.9901	0.6518	0.9500	0.4680	0.7500
0.01	3	3	5	5	0.001	(maxx-minx)/20	0.9841	0.6859	0.9220	0.4960	0.7498
0.0001	2	5	5	10	0.001	(maxx-minx)/20	0.9940	0.6225	0.9720	0.4440	0.7496
0.0001	2	3	3	10	0.001	(maxx-minx)/20	0.9945	0.6268	0.9720	0.4380	0.7494
0.0001	2	3	5	10	0.001	(maxx-minx)/20	0.9919	0.6410	0.9580	0.4560	0.7492
0.0001	2	7	5	10	0.01	(maxx-minx)/20	0.9941	0.6173	0.9720	0.4460	0.7489
0.0001	2	3	7	5	0.001	(maxx-minx)/20	0.9916	0.6463	0.9580	0.4480	0.7484
0.01	2	3	7	7	0.001	(maxx-minx)/2	0.9932	0.6103	0.9680	0.4600	0.7482
0.01	2	5	3	5	0.001	(maxx-minx)/2	0.9949	0.6095	0.9740	0.4440	0.7478
0.01	3	3	7	7	0.01	(maxx-minx)/20	0.9850	0.6671	0.9360	0.4780	0.7478
0.01	2	5	5	7	0.001	(maxx-minx)/2	0.9946	0.5993	0.9740	0.4520	0.7478
0.01	3	7	3	5	0.01	(maxx-minx)/20	0.9923	0.6424	0.9600	0.4440	0.7478
0.01	3	5	3	5	0.01	(maxx-minx)/20	0.9876	0.6633	0.9440	0.4640	0.7476
0.01	2	5	5	10	0.001	(maxx-minx)/2	0.9973	0.5859	0.9860	0.4380	0.7476
0.0001	2	7	5	10	0.001	(maxx-minx)/20	0.9961	0.6072	0.9800	0.4300	0.7473
0.01	3	3	7	5	0.01	(maxx-minx)/20	0.9827	0.6882	0.9180	0.4960	0.7470
0.01	3	3	3	5	0.001	(maxx-minx)/20	0.9812	0.6906	0.9240	0.4800	0.7469
0.0001	2	7	3	5	0.001	(maxx-minx)/20	0.9941	0.6197	0.9720	0.4340	0.7465
0.0001	2	3	7	10	0.001	(maxx-minx)/2	0.9966	0.5786	0.9860	0.4420	0.7464
0.0001	2	5	7	7	0.001	(maxx-minx)/2	0.9984	0.5727	0.9920	0.4320	0.7464
0.0001	2	5	5	7	0.001	(maxx-minx)/20	0.9924	0.6285	0.9640	0.4420	0.7458
0.0001	2	5	7	10	0.001	(maxx-minx)/20	0.9937	0.6156	0.9680	0.4440	0.7458
0.0001	2	7	7	10	0.001	(maxx-minx)/20	0.9945	0.6070	0.9720	0.4420	0.7455
0.01	2	7	5	10	0.001	(maxx-minx)/20	0.9924	0.6367	0.9620	0.4360	0.7454
0.01	3	3	5	10	0.001	(maxx-minx)/20	0.9874	0.6520	0.9360	0.4780	0.7448
0.0001	2	5	3	10	0.001	(maxx-minx)/20	0.9945	0.6101	0.9720	0.4360	0.7448
0.0001	2	5	7	10	0.01	(maxx-minx)/20	0.9923	0.6240	0.9600	0.4500	0.7446
0.01	2	7	7	10	0.001	(maxx-minx)/2	0.9968	0.5806	0.9840	0.4360	0.7445
0.01	2	7	7	7	0.001	(maxx-minx)/2	0.9951	0.5869	0.9760	0.4460	0.7444
0.0001	2	3	3	10	0.001	(maxx-minx)/2	0.9953	0.5827	0.9780	0.4480	0.7443
0.01	2	3	7	5	0.001	(maxx-minx)/2	0.9927	0.6091	0.9640	0.4540	0.7441
0.01	3	5	5	7	0.01	(maxx-minx)/20	0.9907	0.6400	0.9540	0.4440	0.7434
0.0001	2	5	3	7	0.001	(maxx-minx)/20	0.9925	0.6281	0.9620	0.4360	0.7433
0.01	2	3	7	10	0.001	(maxx-minx)/2	0.9909	0.6084	0.9600	0.4620	0.7430
0.0001	2	5	5	10	0.001	(maxx-minx)/2	0.9980	0.5680	0.9900	0.4280	0.7430
0.0001	2	3	3	5	0.001	(maxx-minx)/2	0.9965	0.5825	0.9820	0.4320	0.7429
0.01	3	5	7	5	0.01	(maxx-minx)/20	0.9845	0.6657	0.9260	0.4820	0.7428
0.0001	2	3	5	10	0.001	(maxx-minx)/2	0.9949	0.5803	0.9760	0.4480	0.7425
0.01	2	7	7	5	0.001	(maxx-minx)/2	0.9970	0.5798	0.9840	0.4280	0.7425
0.01	3	5	3	5	0.001	(maxx-minx)/20	0.9866	0.6601	0.9340	0.4680	0.7423
0.01	3	3	7	10	0.01	(maxx-minx)/20	0.9846	0.6619	0.9280	0.4820	0.7423

0.0001	2	5	3	5	0.001	(maxx-minx)/2	0.9965	0.5797	0.9820	0.4320	0.7422
0.01	2	3	3	7	0.001	(maxx-minx)/2	0.9925	0.6113	0.9640	0.4440	0.7420
0.0001	2	3	7	10	0.01	(maxx-minx)/2	0.9936	0.5809	0.9720	0.4560	0.7420
0.01	2	7	3	10	0.001	(maxx-minx)/2	0.9961	0.5852	0.9800	0.4280	0.7414
0.01	3	3	5	10	0.01	(maxx-minx)/20	0.9848	0.6578	0.9320	0.4700	0.7413
0.01	2	5	3	10	0.001	(maxx-minx)/2	0.9948	0.5974	0.9740	0.4300	0.7412
0.01	3	3	7	7	0.001	(maxx-minx)/20	0.9861	0.6638	0.9320	0.4620	0.7411
0.0001	2	3	5	7	0.001	(maxx-minx)/20	0.9902	0.6396	0.9500	0.4440	0.7410
0.0001	2	7	7	7	0.001	(maxx-minx)/2	0.9951	0.5734	0.9760	0.4480	0.7409
0.01	3	7	5	7	0.01	(maxx-minx)/20	0.9902	0.6303	0.9500	0.4520	0.7407
0.01	2	7	5	5	0.001	(maxx-minx)/2	0.9957	0.5907	0.9780	0.4220	0.7400
0.0001	2	3	7	5	0.001	(maxx-minx)/2	0.9938	0.5859	0.9700	0.4460	0.7399
0.0001	2	7	3	10	0.001	(maxx-minx)/20	0.9929	0.6135	0.9640	0.4320	0.7399
0.0001	2	3	7	7	0.01	(maxx-minx)/2	0.9940	0.5812	0.9720	0.4460	0.7398
0.0001	2	3	5	7	0.001	(maxx-minx)/2	0.9951	0.5799	0.9760	0.4360	0.7395
0.01	3	5	7	5	0.001	(maxx-minx)/20	0.9851	0.6574	0.9280	0.4740	0.7394
0.0001	2	5	5	5	0.001	(maxx-minx)/20	0.9921	0.6240	0.9600	0.4260	0.7392
0.0001	2	7	3	10	0.001	(maxx-minx)/2	0.9969	0.5646	0.9840	0.4300	0.7391
0.0001	2	7	7	10	0.001	(maxx-minx)/2	0.9966	0.5650	0.9840	0.4300	0.7390
0.01	2	5	3	7	0.001	(maxx-minx)/2	0.9936	0.6009	0.9680	0.4320	0.7390
0.0001	2	7	3	7	0.001	(maxx-minx)/2	0.9961	0.5724	0.9800	0.4300	0.7387
0.0001	2	3	5	10	0.01	(maxx-minx)/2	0.9943	0.5753	0.9740	0.4420	0.7385
0.0001	2	3	3	7	0.001	(maxx-minx)/2	0.9943	0.5811	0.9740	0.4360	0.7384
0.0001	2	5	3	10	0.001	(maxx-minx)/2	0.9957	0.5762	0.9800	0.4260	0.7384
0.0001	3	7	3	5	0.001	(maxx-minx)/20	0.9893	0.6175	0.9500	0.4560	0.7380
0.0001	2	5	7	7	0.01	(maxx-minx)/2	0.9952	0.5712	0.9780	0.4340	0.7379
0.01	2	7	5	7	0.001	(maxx-minx)/2	0.9955	0.5801	0.9780	0.4240	0.7378
0.0001	2	7	5	7	0.001	(maxx-minx)/2	0.9970	0.5669	0.9860	0.4180	0.7377
0.01	2	7	5	10	0.001	(maxx-minx)/2	0.9938	0.5868	0.9720	0.4320	0.7376
0.0001	3	3	7	5	0.01	(maxx-minx)/20	0.9884	0.6238	0.9440	0.4600	0.7372
0.0001	2	5	7	10	0.01	(maxx-minx)/2	0.9961	0.5685	0.9820	0.4240	0.7371
0.0001	2	5	7	5	0.01	(maxx-minx)/2	0.9951	0.5703	0.9760	0.4360	0.7371
0.01	3	3	3	7	0.001	(maxx-minx)/20	0.9849	0.6699	0.9240	0.4580	0.7370
0.01	2	5	7	5	0.001	(maxx-minx)/2	0.9920	0.5951	0.9640	0.4380	0.7369
0.01	2	7	3	5	0.001	(maxx-minx)/2	0.9953	0.5839	0.9760	0.4200	0.7366
0.01	3	3	5	7	0.001	(maxx-minx)/20	0.9855	0.6582	0.9300	0.4540	0.7364
0.0001	2	5	5	7	0.001	(maxx-minx)/2	0.9941	0.5773	0.9700	0.4400	0.7364
0.0001	3	3	3	5	0.01	(maxx-minx)/20	0.9887	0.6336	0.9420	0.4500	0.7362
0.0001	2	5	3	7	0.001	(maxx-minx)/2	0.9945	0.5758	0.9740	0.4320	0.7362
0.01	2	5	5	5	0.001	(maxx-minx)/2	0.9943	0.5878	0.9720	0.4240	0.7361
0.01	3	5	5	10	0.01	(maxx-minx)/20	0.9868	0.6400	0.9340	0.4620	0.7359
0.0001	2	5	5	5	0.001	(maxx-minx)/2	0.9933	0.5828	0.9660	0.4420	0.7358
0.0001	2	3	7	5	0.01	(maxx-minx)/2	0.9930	0.5814	0.9680	0.4360	0.7354
0.01	3	5	3	7	0.01	(maxx-minx)/20	0.9873	0.6445	0.9380	0.4440	0.7354
0.01	3	7	5	10	0.01	(maxx-minx)/20	0.9887	0.6236	0.9440	0.4520	0.7352
0.0001	2	5	5	5	0.01	(maxx-minx)/2	0.9906	0.5735	0.9660	0.4500	0.7349
0.0001	3	3	5	5	0.001	(maxx-minx)/20	0.9885	0.6204	0.9460	0.4500	0.7349
0.0001	3	5	5	10	0.001	(maxx-minx)/20	0.9897	0.5998	0.9520	0.4560	0.7348
0.0001	3	5	3	5	0.01	(maxx-minx)/20	0.9897	0.6190	0.9460	0.4460	0.7347
0.01	3	5	7	10	0.001	(maxx-minx)/20	0.9860	0.6318	0.9340	0.4620	0.7347
0.0001	3	7	5	5	0.01	(maxx-minx)/20	0.9897	0.6092	0.9500	0.4500	0.7347

0.0001	2	3	5	5	0.001	(maxx-minx)/2	0.9942	0.5802	0.9720	0.4260	0.7346
0.0001	2	5	5	10	0.01	(maxx-minx)/2	0.9947	0.5652	0.9760	0.4320	0.7346
0.0001	2	5	7	5	0.001	(maxx-minx)/2	0.9930	0.5785	0.9660	0.4380	0.7343
0.01	2	7	3	7	0.001	(maxx-minx)/2	0.9940	0.5825	0.9700	0.4260	0.7341
0.01	3	7	7	5	0.01	(maxx-minx)/20	0.9872	0.6357	0.9360	0.4540	0.7340
0.0001	2	7	5	7	0.01	(maxx-minx)/2	0.9940	0.5649	0.9720	0.4380	0.7337
0.0001	3	5	5	5	0.01	(maxx-minx)/20	0.9894	0.6141	0.9500	0.4420	0.7337
0.0001	3	3	7	10	0.01	(maxx-minx)/20	0.9886	0.6068	0.9500	0.4480	0.7336
0.01	3	7	3	7	0.01	(maxx-minx)/20	0.9889	0.6279	0.9460	0.4360	0.7334
0.0001	2	3	7	7	0.001	(maxx-minx)/2	0.9932	0.5731	0.9680	0.4380	0.7334
0.01	3	7	3	5	0.001	(maxx-minx)/20	0.9871	0.6370	0.9380	0.4460	0.7333
0.01	2	5	7	7	0.001	(maxx-minx)/2	0.9928	0.5892	0.9640	0.4300	0.7332
0.0001	2	7	7	10	0.01	(maxx-minx)/2	0.9918	0.5716	0.9600	0.4560	0.7328
0.0001	2	7	7	5	0.001	(maxx-minx)/2	0.9949	0.5649	0.9740	0.4280	0.7327
0.01	3	7	5	5	0.01	(maxx-minx)/20	0.9861	0.6323	0.9320	0.4620	0.7326
0.0001	2	7	5	5	0.001	(maxx-minx)/2	0.9943	0.5669	0.9720	0.4300	0.7324
0.0001	2	3	5	5	0.01	(maxx-minx)/2	0.9906	0.5819	0.9620	0.4420	0.7323
0.01	3	7	3	10	0.01	(maxx-minx)/20	0.9886	0.6173	0.9440	0.4460	0.7321
0.0001	3	3	5	5	0.01	(maxx-minx)/20	0.9857	0.6324	0.9340	0.4560	0.7320
0.0001	3	3	5	7	0.01	(maxx-minx)/20	0.9884	0.6205	0.9420	0.4440	0.7320
0.01	3	5	7	7	0.01	(maxx-minx)/20	0.9862	0.6411	0.9320	0.4500	0.7318
0.01	3	7	7	7	0.01	(maxx-minx)/20	0.9870	0.6253	0.9380	0.4520	0.7318
0.0001	2	7	5	10	0.001	(maxx-minx)/2	0.9960	0.5645	0.9800	0.4100	0.7316
0.01	3	5	7	10	0.01	(maxx-minx)/20	0.9872	0.6340	0.9360	0.4460	0.7316
0.0001	3	3	3	7	0.01	(maxx-minx)/20	0.9867	0.6229	0.9320	0.4660	0.7316
0.0001	2	7	7	7	0.01	(maxx-minx)/2	0.9929	0.5679	0.9680	0.4360	0.7314
0.0001	3	5	7	7	0.001	(maxx-minx)/20	0.9898	0.5934	0.9500	0.4480	0.7309
0.01	3	7	5	5	0.001	(maxx-minx)/20	0.9876	0.6281	0.9360	0.4480	0.7308
0.0001	3	3	3	10	0.01	(maxx-minx)/20	0.9862	0.6149	0.9340	0.4660	0.7303
0.0001	2	3	5	7	0.01	(maxx-minx)/2	0.9917	0.5734	0.9640	0.4360	0.7302
0.01	3	5	3	7	0.001	(maxx-minx)/20	0.9850	0.6482	0.9280	0.4460	0.7301
0.0001	3	7	7	7	0.01	(maxx-minx)/20	0.9881	0.6064	0.9420	0.4520	0.7297
0.0001	3	3	3	7	0.001	(maxx-minx)/20	0.9878	0.6123	0.9380	0.4520	0.7296
0.0001	3	7	3	10	0.01	(maxx-minx)/20	0.9888	0.5976	0.9460	0.4480	0.7294
0.0001	3	3	7	5	0.001	(maxx-minx)/20	0.9867	0.6197	0.9360	0.4520	0.7293
0.01	2	5	7	10	0.001	(maxx-minx)/2	0.9909	0.5852	0.9580	0.4340	0.7292
0.0001	3	5	7	7	0.01	(maxx-minx)/20	0.9887	0.6054	0.9420	0.4500	0.7292
0.01	3	5	5	7	0.001	(maxx-minx)/20	0.9860	0.6366	0.9300	0.4480	0.7292
0.01	3	7	5	7	0.001	(maxx-minx)/20	0.9881	0.6196	0.9420	0.4360	0.7289
0.01	3	5	5	10	0.001	(maxx-minx)/20	0.9868	0.6256	0.9320	0.4520	0.7288
0.0001	3	3	5	10	0.001	(maxx-minx)/20	0.9885	0.6040	0.9420	0.4500	0.7287
0.0001	3	7	7	5	0.001	(maxx-minx)/20	0.9877	0.6045	0.9400	0.4500	0.7287
0.0001	3	7	5	7	0.001	(maxx-minx)/20	0.9904	0.5954	0.9520	0.4340	0.7286
0.0001	2	5	7	10	0.001	(maxx-minx)/2	0.9946	0.5627	0.9720	0.4180	0.7285
0.0001	3	3	7	7	0.01	(maxx-minx)/20	0.9874	0.6190	0.9360	0.4480	0.7284
0.0001	3	5	7	10	0.01	(maxx-minx)/20	0.9881	0.5975	0.9440	0.4500	0.7279
0.0001	3	5	3	7	0.01	(maxx-minx)/20	0.9877	0.6082	0.9400	0.4480	0.7279
0.0001	3	7	7	7	0.001	(maxx-minx)/20	0.9884	0.5982	0.9440	0.4460	0.7279
0.0001	3	7	5	10	0.01	(maxx-minx)/20	0.9886	0.5993	0.9420	0.4500	0.7276
0.0001	3	5	5	7	0.01	(maxx-minx)/20	0.9869	0.6183	0.9360	0.4460	0.7275
0.0001	3	7	7	5	0.01	(maxx-minx)/20	0.9885	0.6068	0.9420	0.4420	0.7274

0.0001	3	5	3	5	0.001	(maxx-minx)/20	0.9883	0.6069	0.9420	0.4420	0.7274
0.0001	3	3	3	5	0.001	(maxx-minx)/20	0.9842	0.6287	0.9240	0.4640	0.7273
0.0001	2	7	7	5	0.01	(maxx-minx)/2	0.9919	0.5751	0.9600	0.4300	0.7272
0.01	3	5	3	10	0.001	(maxx-minx)/20	0.9860	0.6250	0.9300	0.4480	0.7269
0.01	3	7	7	10	0.01	(maxx-minx)/20	0.9868	0.6168	0.9360	0.4420	0.7267
0.0001	2	7	3	5	0.001	(maxx-minx)/2	0.9907	0.5711	0.9560	0.4360	0.7258
0.01	3	7	7	10	0.001	(maxx-minx)/20	0.9864	0.6114	0.9360	0.4420	0.7252
0.01	3	7	7	7	0.001	(maxx-minx)/20	0.9869	0.6169	0.9360	0.4380	0.7252
0.0001	3	5	5	5	0.001	(maxx-minx)/20	0.9878	0.6068	0.9380	0.4380	0.7247
0.0001	3	5	7	5	0.001	(maxx-minx)/20	0.9867	0.6054	0.9400	0.4400	0.7247
0.0001	3	5	7	5	0.01	(maxx-minx)/20	0.9875	0.6052	0.9400	0.4380	0.7246
0.01	3	5	3	10	0.01	(maxx-minx)/20	0.9840	0.6341	0.9200	0.4560	0.7245
0.0001	2	3	3	7	0.01	(maxx-minx)/2	0.9868	0.5703	0.9520	0.4440	0.7245
0.0001	3	7	5	7	0.01	(maxx-minx)/20	0.9884	0.5965	0.9420	0.4380	0.7239
0.01	3	7	5	10	0.001	(maxx-minx)/20	0.9854	0.6145	0.9320	0.4460	0.7238
0.0001	2	5	5	7	0.01	(maxx-minx)/2	0.9893	0.5629	0.9580	0.4320	0.7231
0.0001	3	7	3	5	0.01	(maxx-minx)/20	0.9855	0.6133	0.9300	0.4480	0.7231
0.01	3	7	7	5	0.001	(maxx-minx)/20	0.9839	0.6321	0.9200	0.4460	0.7227
0.0001	2	5	3	10	0.01	(maxx-minx)/2	0.9859	0.5721	0.9500	0.4440	0.7227
0.0001	3	7	3	7	0.01	(maxx-minx)/20	0.9879	0.5988	0.9400	0.4360	0.7227
0.0001	3	3	5	10	0.01	(maxx-minx)/20	0.9854	0.6082	0.9280	0.4540	0.7222
0.0001	3	5	3	10	0.01	(maxx-minx)/20	0.9875	0.5971	0.9420	0.4320	0.7220
0.0001	3	3	3	10	0.001	(maxx-minx)/20	0.9865	0.6026	0.9360	0.4400	0.7219
0.01	4	3	5	5	0.01	(maxx-minx)/20	0.9757	0.6560	0.8940	0.4840	0.7217
0.01	3	3	7	10	0.001	(maxx-minx)/20	0.9817	0.6494	0.9060	0.4620	0.7217
0.0001	3	5	5	7	0.001	(maxx-minx)/20	0.9876	0.5996	0.9400	0.4300	0.7212
0.0001	3	7	5	5	0.001	(maxx-minx)/20	0.9859	0.6080	0.9300	0.4440	0.7210
0.0001	3	3	7	10	0.001	(maxx-minx)/20	0.9851	0.6048	0.9300	0.4460	0.7209
0.01	3	7	3	7	0.001	(maxx-minx)/20	0.9828	0.6271	0.9200	0.4480	0.7202
0.0001	2	3	3	5	0.01	(maxx-minx)/2	0.9831	0.5765	0.9380	0.4500	0.7201
0.0001	2	5	3	5	0.01	(maxx-minx)/2	0.9829	0.5681	0.9460	0.4340	0.7195
0.0001	2	7	5	5	0.01	(maxx-minx)/2	0.9885	0.5693	0.9480	0.4340	0.7191
0.01	3	7	3	10	0.001	(maxx-minx)/20	0.9850	0.6032	0.9300	0.4400	0.7189
0.0001	3	7	7	10	0.01	(maxx-minx)/20	0.9867	0.5890	0.9320	0.4460	0.7181
0.0001	3	7	3	10	0.001	(maxx-minx)/20	0.9869	0.5819	0.9360	0.4440	0.7179
0.0001	3	7	5	10	0.001	(maxx-minx)/20	0.9865	0.5803	0.9380	0.4360	0.7176
0.01	4	5	3	5	0.01	(maxx-minx)/20	0.9800	0.6396	0.9080	0.4520	0.7175
0.0001	2	7	3	7	0.01	(maxx-minx)/2	0.9818	0.5666	0.9420	0.4420	0.7171
0.0001	3	3	7	7	0.001	(maxx-minx)/20	0.9843	0.6071	0.9220	0.4480	0.7169
0.01	3	5	7	7	0.001	(maxx-minx)/20	0.9817	0.6282	0.9120	0.4520	0.7169
0.0001	3	7	7	10	0.001	(maxx-minx)/20	0.9881	0.5804	0.9400	0.4280	0.7168
0.0001	3	5	5	10	0.01	(maxx-minx)/20	0.9845	0.6036	0.9260	0.4420	0.7166
0.0001	2	7	5	10	0.01	(maxx-minx)/2	0.9870	0.5665	0.9460	0.4300	0.7164
0.01	3	7	5	10	0.001	(maxx-minx)/2	0.9893	0.5564	0.9480	0.4320	0.7164
0.0001	3	3	7	10	0.01	(maxx-minx)/2	0.9857	0.5584	0.9400	0.4460	0.7159
0.0001	2	5	3	7	0.01	(maxx-minx)/2	0.9823	0.5697	0.9380	0.4440	0.7154
0.01	4	5	5	5	0.01	(maxx-minx)/20	0.9775	0.6342	0.8980	0.4640	0.7148
0.0001	3	5	5	10	0.001	(maxx-minx)/2	0.9886	0.5584	0.9440	0.4320	0.7145
0.0001	3	7	7	5	0.01	(maxx-minx)/2	0.9869	0.5583	0.9420	0.4380	0.7142
0.01	4	5	7	5	0.01	(maxx-minx)/20	0.9799	0.6257	0.9080	0.4500	0.7141
0.0001	3	3	7	5	0.01	(maxx-minx)/2	0.9866	0.5579	0.9380	0.4440	0.7140

0.0001	3	3	5	10	0.001	(maxx-minx)/2	0.9889	0.5516	0.9460	0.4320	0.7139
0.0001	3	3	7	10	0.001	(maxx-minx)/2	0.9875	0.5607	0.9420	0.4360	0.7139
0.0001	3	3	5	7	0.001	(maxx-minx)/2	0.9862	0.5604	0.9380	0.4460	0.7137
0.01	4	7	5	5	0.01	(maxx-minx)/20	0.9776	0.6302	0.9020	0.4600	0.7136
0.01	3	3	5	7	0.001	(maxx-minx)/2	0.9848	0.5743	0.9280	0.4500	0.7131
0.01	4	5	3	10	0.01	(maxx-minx)/20	0.9792	0.6214	0.9060	0.4580	0.7131
0.0001	2	3	3	10	0.01	(maxx-minx)/2	0.9826	0.5748	0.9320	0.4420	0.7130
0.0001	3	7	7	7	0.01	(maxx-minx)/2	0.9873	0.5540	0.9400	0.4420	0.7127
0.01	3	3	5	10	0.001	(maxx-minx)/2	0.9877	0.5671	0.9400	0.4280	0.7126
0.01	4	3	5	5	0.001	(maxx-minx)/20	0.9752	0.6506	0.8900	0.4580	0.7123
0.01	3	7	7	7	0.001	(maxx-minx)/2	0.9871	0.5595	0.9420	0.4280	0.7120
0.0001	2	7	3	5	0.01	(maxx-minx)/2	0.9831	0.5668	0.9360	0.4360	0.7117
0.0001	3	7	7	10	0.01	(maxx-minx)/2	0.9873	0.5513	0.9400	0.4380	0.7116
0.0001	3	5	3	7	0.001	(maxx-minx)/20	0.9837	0.6005	0.9220	0.4320	0.7116
0.0001	3	5	3	10	0.001	(maxx-minx)/20	0.9859	0.5875	0.9280	0.4300	0.7113
0.0001	3	3	3	10	0.001	(maxx-minx)/2	0.9871	0.5569	0.9380	0.4380	0.7113
0.01	3	7	3	10	0.001	(maxx-minx)/2	0.9882	0.5557	0.9440	0.4200	0.7100
0.01	3	5	7	5	0.001	(maxx-minx)/2	0.9864	0.5649	0.9340	0.4320	0.7100
0.01	4	5	5	5	0.001	(maxx-minx)/20	0.9753	0.6311	0.8920	0.4720	0.7094
0.0001	3	7	3	7	0.001	(maxx-minx)/20	0.9856	0.5879	0.9280	0.4220	0.7093
0.0001	3	7	7	5	0.001	(maxx-minx)/2	0.9885	0.5484	0.9420	0.4260	0.7088
0.01	3	3	5	5	0.001	(maxx-minx)/2	0.9821	0.5746	0.9200	0.4500	0.7085
0.0001	3	5	7	10	0.001	(maxx-minx)/20	0.9852	0.5794	0.9300	0.4240	0.7084
0.01	4	7	3	5	0.01	(maxx-minx)/20	0.9754	0.6357	0.8860	0.4700	0.7084
0.01	3	3	3	10	0.001	(maxx-minx)/20	0.9750	0.6529	0.8820	0.4640	0.7083
0.0001	4	5	7	5	0.01	(maxx-minx)/20	0.9824	0.5950	0.9140	0.4420	0.7081
0.01	4	3	5	7	0.001	(maxx-minx)/20	0.9763	0.6388	0.8880	0.4620	0.7080
0.01	3	5	5	10	0.001	(maxx-minx)/2	0.9865	0.5636	0.9340	0.4260	0.7077
0.0001	3	5	3	10	0.001	(maxx-minx)/2	0.9879	0.5484	0.9420	0.4220	0.7076
0.01	4	5	5	7	0.01	(maxx-minx)/20	0.9769	0.6223	0.9000	0.4480	0.7073
0.0001	3	3	5	7	0.001	(maxx-minx)/20	0.9820	0.6031	0.9120	0.4380	0.7073
0.01	4	5	3	5	0.001	(maxx-minx)/20	0.9750	0.6357	0.8880	0.4620	0.7072
0.0001	3	3	7	7	0.001	(maxx-minx)/2	0.9856	0.5560	0.9340	0.4280	0.7071
0.01	3	5	5	5	0.001	(maxx-minx)/2	0.9828	0.5698	0.9200	0.4420	0.7070
0.0001	4	3	7	7	0.001	(maxx-minx)/20	0.9792	0.6087	0.8980	0.4640	0.7068
0.01	4	3	3	5	0.01	(maxx-minx)/20	0.9701	0.6691	0.8600	0.4920	0.7066
0.0001	3	3	7	5	0.001	(maxx-minx)/2	0.9857	0.5587	0.9340	0.4280	0.7065
0.01	3	7	3	7	0.001	(maxx-minx)/2	0.9848	0.5663	0.9320	0.4260	0.7065
0.0001	3	7	5	10	0.001	(maxx-minx)/2	0.9873	0.5468	0.9400	0.4240	0.7063
0.0001	3	7	3	10	0.001	(maxx-minx)/2	0.9883	0.5494	0.9420	0.4140	0.7060
0.0001	4	7	5	10	0.01	(maxx-minx)/20	0.9793	0.5973	0.9020	0.4620	0.7055
0.01	3	3	3	5	0.001	(maxx-minx)/2	0.9836	0.5738	0.9220	0.4340	0.7054
0.01	4	3	7	5	0.01	(maxx-minx)/20	0.9710	0.6533	0.8740	0.4680	0.7051
0.01	3	5	3	10	0.001	(maxx-minx)/2	0.9852	0.5654	0.9300	0.4220	0.7051
0.01	3	5	7	7	0.001	(maxx-minx)/2	0.9838	0.5634	0.9260	0.4340	0.7049
0.0001	3	3	3	7	0.001	(maxx-minx)/2	0.9831	0.5616	0.9260	0.4380	0.7044
0.0001	4	3	3	5	0.01	(maxx-minx)/20	0.9769	0.6136	0.8900	0.4620	0.7042
0.01	3	3	3	7	0.001	(maxx-minx)/2	0.9789	0.5849	0.9060	0.4540	0.7041
0.0001	3	7	5	5	0.001	(maxx-minx)/2	0.9855	0.5510	0.9320	0.4300	0.7040
0.01	3	3	7	5	0.001	(maxx-minx)/2	0.9809	0.5752	0.9120	0.4440	0.7032
0.01	3	7	7	10	0.001	(maxx-minx)/2	0.9855	0.5572	0.9300	0.4240	0.7031

0.0001	4	7	7	5	0.01	(maxx-minx)/20	0.9802	0.5881	0.9080	0.4420	0.7029
0.0001	4	5	5	5	0.01	(maxx-minx)/20	0.9796	0.6035	0.9000	0.4460	0.7028
0.0001	4	7	5	5	0.01	(maxx-minx)/20	0.9796	0.5989	0.9020	0.4460	0.7026
0.0001	3	5	7	7	0.01	(maxx-minx)/2	0.9842	0.5553	0.9240	0.4360	0.7026
0.0001	4	3	7	7	0.01	(maxx-minx)/20	0.9817	0.5919	0.9120	0.4300	0.7023
0.01	4	3	3	10	0.01	(maxx-minx)/20	0.9735	0.6387	0.8780	0.4620	0.7022
0.01	3	3	7	7	0.001	(maxx-minx)/2	0.9833	0.5646	0.9180	0.4380	0.7019
0.01	4	7	3	7	0.01	(maxx-minx)/20	0.9777	0.6198	0.8920	0.4480	0.7018
0.0001	3	3	5	10	0.01	(maxx-minx)/2	0.9840	0.5542	0.9300	0.4240	0.7016
0.0001	3	7	7	7	0.001	(maxx-minx)/2	0.9852	0.5497	0.9320	0.4220	0.7015
0.0001	3	7	3	7	0.001	(maxx-minx)/2	0.9849	0.5514	0.9280	0.4260	0.7014
0.0001	4	7	3	7	0.01	(maxx-minx)/20	0.9786	0.5910	0.9040	0.4420	0.7014
0.01	4	3	3	5	0.001	(maxx-minx)/20	0.9701	0.6579	0.8700	0.4580	0.7009
0.0001	4	7	7	7	0.001	(maxx-minx)/20	0.9798	0.5880	0.9040	0.4480	0.7009
0.0001	4	7	7	7	0.01	(maxx-minx)/20	0.9804	0.5855	0.9040	0.4440	0.7008
0.0001	3	7	5	7	0.001	(maxx-minx)/2	0.9857	0.5491	0.9320	0.4140	0.7002
0.0001	4	5	3	5	0.01	(maxx-minx)/20	0.9785	0.6013	0.9020	0.4380	0.7001
0.0001	3	7	7	10	0.001	(maxx-minx)/2	0.9851	0.5455	0.9300	0.4240	0.6999
0.0001	3	5	5	7	0.01	(maxx-minx)/2	0.9802	0.5579	0.9140	0.4500	0.6998
0.0001	4	3	5	7	0.01	(maxx-minx)/20	0.9773	0.6057	0.8920	0.4540	0.6996
0.0001	3	3	5	7	0.01	(maxx-minx)/2	0.9835	0.5511	0.9260	0.4280	0.6995
0.0001	3	5	7	10	0.001	(maxx-minx)/2	0.9849	0.5537	0.9260	0.4200	0.6995
0.0001	4	7	5	7	0.01	(maxx-minx)/20	0.9797	0.5902	0.9000	0.4460	0.6995
0.0001	3	5	7	5	0.01	(maxx-minx)/2	0.9817	0.5531	0.9220	0.4320	0.6994
0.01	4	5	3	10	0.001	(maxx-minx)/20	0.9745	0.6180	0.8800	0.4680	0.6994
0.0001	4	3	3	10	0.01	(maxx-minx)/20	0.9794	0.5923	0.8980	0.4420	0.6992
0.01	3	5	5	7	0.001	(maxx-minx)/2	0.9829	0.5561	0.9220	0.4180	0.6991
0.01	3	7	5	7	0.001	(maxx-minx)/2	0.9850	0.5557	0.9280	0.4140	0.6990
0.01	4	7	5	5	0.001	(maxx-minx)/20	0.9750	0.6262	0.8860	0.4460	0.6986
0.0001	4	7	3	5	0.01	(maxx-minx)/20	0.9793	0.5982	0.8960	0.4420	0.6983
0.01	4	5	7	7	0.01	(maxx-minx)/20	0.9760	0.6198	0.8880	0.4440	0.6980
0.01	4	7	7	7	0.01	(maxx-minx)/20	0.9735	0.6117	0.8860	0.4560	0.6979
0.0001	4	3	5	5	0.01	(maxx-minx)/20	0.9772	0.6066	0.8880	0.4520	0.6979
0.0001	2	7	3	10	0.01	(maxx-minx)/2	0.9751	0.5693	0.9100	0.4400	0.6979
0.0001	3	5	5	5	0.001	(maxx-minx)/2	0.9843	0.5515	0.9220	0.4220	0.6978
0.01	4	5	7	5	0.001	(maxx-minx)/20	0.9760	0.6191	0.8860	0.4480	0.6978
0.0001	3	5	7	10	0.01	(maxx-minx)/2	0.9800	0.5604	0.9100	0.4480	0.6977
0.01	3	5	7	10	0.001	(maxx-minx)/2	0.9828	0.5558	0.9200	0.4260	0.6975
0.0001	3	5	7	5	0.001	(maxx-minx)/2	0.9839	0.5508	0.9200	0.4260	0.6974
0.0001	3	5	7	7	0.001	(maxx-minx)/2	0.9832	0.5517	0.9240	0.4220	0.6970
0.01	3	3	7	10	0.001	(maxx-minx)/2	0.9812	0.5636	0.9140	0.4340	0.6970
0.01	4	7	7	7	0.001	(maxx-minx)/20	0.9796	0.6026	0.9000	0.4260	0.6970
0.0001	3	3	3	5	0.001	(maxx-minx)/2	0.9828	0.5592	0.9160	0.4280	0.6968
0.0001	3	3	5	5	0.001	(maxx-minx)/2	0.9813	0.5608	0.9180	0.4280	0.6968
0.0001	3	5	3	5	0.001	(maxx-minx)/2	0.9836	0.5575	0.9200	0.4200	0.6968
0.01	4	7	7	5	0.01	(maxx-minx)/20	0.9759	0.6160	0.8900	0.4340	0.6967
0.01	3	5	3	7	0.001	(maxx-minx)/2	0.9820	0.5643	0.9160	0.4200	0.6963
0.01	3	7	5	5	0.001	(maxx-minx)/2	0.9827	0.5614	0.9200	0.4160	0.6963
0.01	4	7	5	10	0.01	(maxx-minx)/20	0.9779	0.6008	0.8920	0.4420	0.6963
0.0001	4	5	5	7	0.01	(maxx-minx)/20	0.9750	0.6037	0.8840	0.4600	0.6961
0.01	4	3	3	7	0.01	(maxx-minx)/20	0.9700	0.6464	0.8600	0.4700	0.6960

0.01	3	7	3	5	0.001	(maxx-minx)/2	0.9790	0.5688	0.9080	0.4380	0.6959
0.0001	4	7	3	10	0.01	(maxx-minx)/20	0.9772	0.5892	0.8900	0.4560	0.6955
0.0001	3	5	5	10	0.01	(maxx-minx)/2	0.9812	0.5455	0.9180	0.4380	0.6955
0.0001	4	3	7	10	0.01	(maxx-minx)/20	0.9774	0.5945	0.8920	0.4460	0.6954
0.01	4	5	3	7	0.01	(maxx-minx)/20	0.9743	0.6238	0.8800	0.4440	0.6954
0.01	3	3	3	10	0.001	(maxx-minx)/2	0.9822	0.5707	0.9120	0.4220	0.6953
0.0001	3	5	5	7	0.001	(maxx-minx)/2	0.9839	0.5488	0.9220	0.4180	0.6952
0.01	3	7	7	5	0.001	(maxx-minx)/2	0.9838	0.5599	0.9200	0.4120	0.6949
0.0001	4	5	3	5	0.001	(maxx-minx)/20	0.9750	0.5969	0.8880	0.4460	0.6947
0.0001	3	5	3	7	0.001	(maxx-minx)/2	0.9826	0.5504	0.9200	0.4220	0.6944
0.0001	4	7	7	5	0.001	(maxx-minx)/20	0.9752	0.5982	0.8840	0.4600	0.6941
0.0001	4	5	3	7	0.01	(maxx-minx)/20	0.9778	0.5977	0.8900	0.4400	0.6939
0.01	4	7	3	10	0.01	(maxx-minx)/20	0.9748	0.6159	0.8800	0.4500	0.6939
0.0001	3	7	5	5	0.01	(maxx-minx)/2	0.9794	0.5481	0.9160	0.4280	0.6937
0.0001	4	5	5	7	0.001	(maxx-minx)/20	0.9761	0.5926	0.8880	0.4500	0.6933
0.0001	4	7	5	7	0.001	(maxx-minx)/20	0.9779	0.5828	0.8940	0.4440	0.6933
0.01	4	3	5	7	0.01	(maxx-minx)/20	0.9713	0.6348	0.8680	0.4580	0.6929
0.01	4	3	7	5	0.001	(maxx-minx)/20	0.9696	0.6470	0.8580	0.4660	0.6927
0.01	4	3	7	7	0.01	(maxx-minx)/20	0.9686	0.6371	0.8660	0.4620	0.6921
0.0001	4	7	7	10	0.01	(maxx-minx)/20	0.9765	0.5806	0.8980	0.4360	0.6920
0.01	4	3	7	10	0.01	(maxx-minx)/20	0.9708	0.6229	0.8680	0.4640	0.6918
0.0001	3	7	5	10	0.01	(maxx-minx)/2	0.9792	0.5520	0.9140	0.4260	0.6917
0.0001	3	7	3	5	0.001	(maxx-minx)/2	0.9811	0.5518	0.9160	0.4200	0.6915
0.01	4	5	5	10	0.01	(maxx-minx)/20	0.9744	0.6191	0.8760	0.4460	0.6915
0.0001	4	3	5	5	0.001	(maxx-minx)/20	0.9744	0.6065	0.8800	0.4440	0.6911
0.01	3	5	3	5	0.001	(maxx-minx)/2	0.9799	0.5711	0.9000	0.4320	0.6907
0.0001	4	3	5	7	0.001	(maxx-minx)/20	0.9732	0.6017	0.8740	0.4600	0.6903
0.01	4	7	5	7	0.01	(maxx-minx)/20	0.9733	0.6145	0.8820	0.4360	0.6903
0.0001	4	5	7	10	0.01	(maxx-minx)/20	0.9749	0.5941	0.8800	0.4520	0.6896
0.0001	4	3	3	7	0.01	(maxx-minx)/20	0.9751	0.5975	0.8840	0.4400	0.6895
0.0001	4	3	7	5	0.01	(maxx-minx)/20	0.9724	0.6047	0.8760	0.4560	0.6894
0.0001	4	3	3	7	0.001	(maxx-minx)/20	0.9731	0.5999	0.8760	0.4540	0.6892
0.0001	4	3	5	10	0.01	(maxx-minx)/20	0.9741	0.5953	0.8840	0.4420	0.6890
0.0001	3	5	3	7	0.01	(maxx-minx)/2	0.9759	0.5500	0.9100	0.4300	0.6887
0.01	4	3	3	7	0.001	(maxx-minx)/20	0.9646	0.6452	0.8480	0.4740	0.6883
0.01	4	7	3	7	0.001	(maxx-minx)/20	0.9722	0.6146	0.8720	0.4500	0.6882
0.01	4	5	5	7	0.001	(maxx-minx)/20	0.9696	0.6221	0.8700	0.4500	0.6878
0.0001	4	5	5	10	0.01	(maxx-minx)/20	0.9744	0.5946	0.8800	0.4480	0.6878
0.0001	3	3	3	10	0.01	(maxx-minx)/2	0.9733	0.5541	0.9000	0.4380	0.6877
0.0001	3	7	3	10	0.01	(maxx-minx)/2	0.9751	0.5519	0.9020	0.4360	0.6875
0.0001	4	3	3	7	0.001	(maxx-minx)/2	0.9758	0.5613	0.8960	0.4340	0.6874
0.0001	4	7	3	5	0.001	(maxx-minx)/20	0.9730	0.5935	0.8840	0.4400	0.6868
0.0001	4	3	3	5	0.001	(maxx-minx)/20	0.9729	0.6081	0.8720	0.4460	0.6866
0.01	4	3	7	10	0.001	(maxx-minx)/2	0.9748	0.5642	0.8900	0.4460	0.6862
0.01	4	3	7	10	0.001	(maxx-minx)/20	0.9677	0.6283	0.8620	0.4520	0.6856
0.0001	4	5	7	5	0.001	(maxx-minx)/20	0.9742	0.5887	0.8800	0.4400	0.6855
0.01	4	5	5	10	0.001	(maxx-minx)/20	0.9704	0.6118	0.8660	0.4420	0.6854
0.0001	4	5	7	7	0.001	(maxx-minx)/20	0.9739	0.5859	0.8840	0.4380	0.6849
0.01	4	7	5	7	0.001	(maxx-minx)/20	0.9715	0.6136	0.8660	0.4480	0.6848
0.0001	4	5	3	10	0.01	(maxx-minx)/20	0.9733	0.5954	0.8760	0.4440	0.6845
0.0001	3	3	5	5	0.01	(maxx-minx)/2	0.9774	0.5546	0.8980	0.4320	0.6844

0.0001	4	5	3	7	0.001	(maxx-minx)/20	0.9697	0.5967	0.8740	0.4480	0.6843
0.01	4	3	5	10	0.01	(maxx-minx)/20	0.9692	0.6297	0.8560	0.4560	0.6841
0.0001	4	5	7	7	0.01	(maxx-minx)/20	0.9752	0.5877	0.8780	0.4420	0.6840
0.0001	4	7	5	5	0.001	(maxx-minx)/20	0.9755	0.5825	0.8820	0.4380	0.6839
0.0001	3	7	5	7	0.01	(maxx-minx)/2	0.9773	0.5441	0.9040	0.4260	0.6832
0.0001	4	3	7	5	0.001	(maxx-minx)/20	0.9728	0.6002	0.8640	0.4540	0.6832
0.01	4	5	7	10	0.001	(maxx-minx)/20	0.9722	0.6012	0.8720	0.4400	0.6831
0.0001	3	5	5	5	0.01	(maxx-minx)/2	0.9743	0.5557	0.8920	0.4400	0.6829
0.01	4	7	5	5	0.001	(maxx-minx)/2	0.9771	0.5573	0.8920	0.4320	0.6825
0.01	4	5	5	7	0.001	(maxx-minx)/2	0.9763	0.5580	0.8940	0.4280	0.6823
0.0001	4	3	7	10	0.001	(maxx-minx)/20	0.9738	0.5832	0.8740	0.4400	0.6822
0.01	4	5	3	7	0.001	(maxx-minx)/20	0.9678	0.6205	0.8620	0.4460	0.6822
0.01	4	3	3	10	0.001	(maxx-minx)/20	0.9610	0.6416	0.8420	0.4760	0.6819
0.0001	4	5	5	5	0.001	(maxx-minx)/20	0.9738	0.5895	0.8740	0.4400	0.6819
0.0001	3	5	3	10	0.01	(maxx-minx)/2	0.9689	0.5520	0.8920	0.4360	0.6818
0.01	4	7	3	5	0.001	(maxx-minx)/20	0.9711	0.6215	0.8620	0.4360	0.6816
0.0001	4	5	7	10	0.001	(maxx-minx)/20	0.9723	0.5824	0.8740	0.4460	0.6815
0.01	4	7	7	10	0.01	(maxx-minx)/20	0.9749	0.5877	0.8800	0.4240	0.6810
0.01	4	5	7	7	0.001	(maxx-minx)/20	0.9691	0.6129	0.8580	0.4480	0.6807
0.0001	4	7	7	7	0.001	(maxx-minx)/2	0.9778	0.5455	0.8940	0.4260	0.6807
0.0001	4	7	7	10	0.001	(maxx-minx)/2	0.9765	0.5448	0.8960	0.4220	0.6805
0.01	4	3	7	7	0.001	(maxx-minx)/20	0.9673	0.6326	0.8480	0.4520	0.6800
0.01	4	5	5	10	0.001	(maxx-minx)/2	0.9775	0.5460	0.8940	0.4260	0.6800
0.0001	4	7	5	10	0.01	(maxx-minx)/2	0.9717	0.5468	0.8920	0.4280	0.6798
0.0001	4	7	7	10	0.001	(maxx-minx)/20	0.9742	0.5783	0.8760	0.4380	0.6798
0.01	4	7	3	10	0.001	(maxx-minx)/20	0.9707	0.5985	0.8680	0.4380	0.6797
0.0001	4	5	5	10	0.001	(maxx-minx)/20	0.9745	0.5807	0.8740	0.4400	0.6794
0.0001	4	5	7	10	0.001	(maxx-minx)/2	0.9765	0.5451	0.8960	0.4220	0.6794
0.0001	4	3	5	10	0.001	(maxx-minx)/20	0.9717	0.5893	0.8680	0.4420	0.6783
0.01	4	7	3	10	0.001	(maxx-minx)/2	0.9756	0.5519	0.8900	0.4200	0.6776
0.0001	3	5	3	5	0.01	(maxx-minx)/2	0.9727	0.5417	0.8980	0.4180	0.6774
0.0001	4	3	3	10	0.001	(maxx-minx)/2	0.9738	0.5500	0.8880	0.4280	0.6773
0.0001	3	7	3	7	0.01	(maxx-minx)/2	0.9717	0.5505	0.8920	0.4220	0.6773
0.0001	4	7	7	10	0.01	(maxx-minx)/2	0.9769	0.5430	0.8920	0.4220	0.6770
0.01	4	7	7	10	0.001	(maxx-minx)/2	0.9748	0.5538	0.8840	0.4280	0.6767
0.0001	4	5	7	10	0.01	(maxx-minx)/2	0.9737	0.5456	0.8900	0.4260	0.6767
0.0001	3	3	7	7	0.01	(maxx-minx)/2	0.9744	0.5442	0.8920	0.4220	0.6766
0.0001	4	5	5	10	0.01	(maxx-minx)/2	0.9746	0.5469	0.8900	0.4240	0.6763
0.01	4	7	3	7	0.001	(maxx-minx)/2	0.9731	0.5568	0.8840	0.4240	0.6758
0.01	4	5	7	10	0.01	(maxx-minx)/20	0.9682	0.6057	0.8600	0.4380	0.6756
0.0001	4	5	3	10	0.001	(maxx-minx)/20	0.9724	0.5829	0.8660	0.4400	0.6756
0.0001	4	7	3	10	0.001	(maxx-minx)/20	0.9722	0.5813	0.8640	0.4480	0.6754
0.0001	4	7	5	10	0.001	(maxx-minx)/20	0.9732	0.5754	0.8740	0.4300	0.6750
0.01	4	5	3	7	0.001	(maxx-minx)/2	0.9723	0.5631	0.8740	0.4340	0.6749
0.01	4	7	5	7	0.001	(maxx-minx)/2	0.9740	0.5594	0.8840	0.4240	0.6748
0.0001	4	5	5	7	0.001	(maxx-minx)/2	0.9765	0.5489	0.8860	0.4200	0.6748
0.01	2	3	7	7	0.01	(maxx-minx)/2	0.9285	0.6133	0.8420	0.4720	0.6741
0.0001	4	5	7	7	0.001	(maxx-minx)/2	0.9755	0.5468	0.8820	0.4260	0.6738
0.01	4	7	7	5	0.001	(maxx-minx)/20	0.9680	0.6182	0.8540	0.4300	0.6737
0.01	2	7	7	5	0.01	(maxx-minx)/2	0.9402	0.5931	0.8520	0.4600	0.6736
0.0001	4	7	5	5	0.001	(maxx-minx)/2	0.9758	0.5464	0.8820	0.4200	0.6730

0.01	2	5	7	7	0.01	(maxx-minx)/2	0.9395	0.5955	0.8580	0.4420	0.6729
0.0001	3	7	3	5	0.01	(maxx-minx)/2	0.9658	0.5477	0.8840	0.4260	0.6728
0.0001	4	3	7	10	0.001	(maxx-minx)/2	0.9735	0.5483	0.8820	0.4240	0.6727
0.01	4	7	7	7	0.001	(maxx-minx)/2	0.9726	0.5571	0.8720	0.4360	0.6724
0.0001	3	3	3	7	0.01	(maxx-minx)/2	0.9705	0.5473	0.8840	0.4240	0.6722
0.01	4	5	7	5	0.001	(maxx-minx)/2	0.9720	0.5605	0.8700	0.4400	0.6717
0.0001	4	5	7	7	0.01	(maxx-minx)/2	0.9711	0.5464	0.8740	0.4320	0.6717
0.0001	4	7	3	7	0.001	(maxx-minx)/20	0.9728	0.5780	0.8660	0.4260	0.6710
0.01	4	3	5	7	0.001	(maxx-minx)/2	0.9675	0.5605	0.8660	0.4440	0.6710
0.0001	4	7	5	7	0.001	(maxx-minx)/2	0.9718	0.5517	0.8720	0.4280	0.6706
0.01	2	3	7	10	0.01	(maxx-minx)/2	0.9297	0.6097	0.8340	0.4740	0.6703
0.01	4	5	7	10	0.001	(maxx-minx)/2	0.9699	0.5566	0.8720	0.4280	0.6703
0.0001	4	7	7	5	0.01	(maxx-minx)/2	0.9709	0.5511	0.8680	0.4420	0.6702
0.0001	4	5	3	5	0.001	(maxx-minx)/2	0.9742	0.5523	0.8740	0.4260	0.6699
0.0001	4	3	3	5	0.001	(maxx-minx)/2	0.9709	0.5526	0.8700	0.4300	0.6699
0.01	2	3	7	5	0.01	(maxx-minx)/2	0.9345	0.6030	0.8460	0.4500	0.6695
0.0001	4	5	3	10	0.001	(maxx-minx)/2	0.9699	0.5512	0.8740	0.4260	0.6694
0.0001	4	3	3	10	0.01	(maxx-minx)/2	0.9641	0.5522	0.8720	0.4380	0.6692
0.01	4	7	5	10	0.001	(maxx-minx)/2	0.9727	0.5535	0.8740	0.4220	0.6691
0.0001	4	3	7	7	0.001	(maxx-minx)/2	0.9707	0.5455	0.8760	0.4220	0.6685
0.01	2	5	7	10	0.01	(maxx-minx)/2	0.9381	0.6004	0.8480	0.4340	0.6679
0.0001	4	7	5	10	0.001	(maxx-minx)/2	0.9718	0.5410	0.8780	0.4200	0.6678
0.0001	4	3	5	10	0.001	(maxx-minx)/2	0.9717	0.5485	0.8700	0.4260	0.6676
0.0001	4	3	3	10	0.001	(maxx-minx)/20	0.9678	0.5852	0.8520	0.4400	0.6675
0.0001	4	5	5	5	0.001	(maxx-minx)/2	0.9676	0.5588	0.8600	0.4380	0.6674
0.0001	4	5	3	7	0.001	(maxx-minx)/2	0.9710	0.5456	0.8700	0.4260	0.6660
0.0001	3	3	3	5	0.01	(maxx-minx)/2	0.9641	0.5524	0.8660	0.4280	0.6659
0.0001	4	5	7	5	0.01	(maxx-minx)/2	0.9692	0.5504	0.8620	0.4340	0.6655
0.01	4	3	5	10	0.001	(maxx-minx)/2	0.9686	0.5578	0.8660	0.4260	0.6651
0.01	4	7	7	10	0.001	(maxx-minx)/20	0.9666	0.5931	0.8480	0.4380	0.6651
0.01	4	7	5	10	0.001	(maxx-minx)/20	0.9666	0.5999	0.8440	0.4360	0.6649
0.0001	4	3	5	7	0.001	(maxx-minx)/2	0.9710	0.5464	0.8760	0.4140	0.6642
0.0001	4	3	7	10	0.01	(maxx-minx)/2	0.9670	0.5464	0.8620	0.4280	0.6638
0.01	2	7	7	7	0.01	(maxx-minx)/2	0.9392	0.5757	0.8560	0.4280	0.6635
0.0001	4	7	5	7	0.01	(maxx-minx)/2	0.9678	0.5390	0.8720	0.4260	0.6634
0.0001	4	5	5	7	0.01	(maxx-minx)/2	0.9678	0.5435	0.8700	0.4260	0.6631
0.01	2	5	7	5	0.01	(maxx-minx)/2	0.9327	0.5924	0.8440	0.4320	0.6630
0.01	4	3	5	10	0.001	(maxx-minx)/20	0.9655	0.6100	0.8360	0.4360	0.6629
0.01	4	5	5	5	0.001	(maxx-minx)/2	0.9688	0.5540	0.8620	0.4280	0.6628
0.0001	4	3	7	5	0.01	(maxx-minx)/2	0.9696	0.5434	0.8680	0.4260	0.6624
0.01	4	3	7	7	0.001	(maxx-minx)/2	0.9682	0.5517	0.8600	0.4300	0.6620
0.01	2	3	5	7	0.01	(maxx-minx)/2	0.9097	0.6162	0.8160	0.4800	0.6619
0.01	4	5	7	7	0.001	(maxx-minx)/2	0.9686	0.5524	0.8600	0.4300	0.6618
0.01	4	3	3	7	0.001	(maxx-minx)/2	0.9669	0.5670	0.8480	0.4380	0.6612
0.01	4	5	3	5	0.001	(maxx-minx)/2	0.9659	0.5541	0.8560	0.4260	0.6610
0.0001	4	7	3	5	0.001	(maxx-minx)/2	0.9674	0.5445	0.8660	0.4220	0.6609
0.01	4	3	5	5	0.001	(maxx-minx)/2	0.9630	0.5581	0.8520	0.4360	0.6607
0.0001	4	5	5	10	0.001	(maxx-minx)/2	0.9681	0.5481	0.8580	0.4280	0.6602
0.01	4	3	3	10	0.001	(maxx-minx)/2	0.9686	0.5610	0.8560	0.4240	0.6598
0.0001	4	3	5	5	0.01	(maxx-minx)/2	0.9660	0.5463	0.8600	0.4300	0.6597
0.0001	4	3	5	5	0.001	(maxx-minx)/2	0.9686	0.5518	0.8560	0.4320	0.6595

0.01	2	7	7	10	0.01	(maxx-minx)/2	0.9330	0.5814	0.8440	0.4340	0.6593
0.0001	4	7	3	10	0.001	(maxx-minx)/2	0.9692	0.5527	0.8580	0.4240	0.6590
0.0001	4	5	7	5	0.001	(maxx-minx)/2	0.9698	0.5495	0.8560	0.4220	0.6589
0.01	2	3	5	5	0.01	(maxx-minx)/2	0.9132	0.6126	0.8220	0.4660	0.6587
0.01	4	5	3	10	0.001	(maxx-minx)/2	0.9667	0.5582	0.8520	0.4240	0.6582
0.0001	4	3	7	7	0.01	(maxx-minx)/2	0.9692	0.5445	0.8580	0.4280	0.6581
0.0001	4	3	5	7	0.01	(maxx-minx)/2	0.9673	0.5436	0.8580	0.4240	0.6578
0.0001	4	7	3	7	0.001	(maxx-minx)/2	0.9670	0.5492	0.8540	0.4200	0.6572
0.0001	4	5	5	5	0.01	(maxx-minx)/2	0.9665	0.5445	0.8540	0.4280	0.6566
0.0001	4	7	7	5	0.001	(maxx-minx)/2	0.9681	0.5496	0.8520	0.4240	0.6566
0.0001	4	5	3	10	0.01	(maxx-minx)/2	0.9595	0.5454	0.8580	0.4260	0.6552
0.01	2	5	5	10	0.01	(maxx-minx)/2	0.9200	0.5906	0.8360	0.4340	0.6551
0.01	2	3	5	10	0.01	(maxx-minx)/2	0.9010	0.6117	0.8200	0.4560	0.6543
0.01	4	7	7	5	0.001	(maxx-minx)/2	0.9656	0.5517	0.8440	0.4240	0.6539
0.01	2	7	5	10	0.01	(maxx-minx)/2	0.9161	0.5845	0.8300	0.4320	0.6537
0.01	4	3	7	5	0.001	(maxx-minx)/2	0.9645	0.5488	0.8440	0.4260	0.6531
0.0001	4	3	7	5	0.001	(maxx-minx)/2	0.9674	0.5469	0.8500	0.4220	0.6528
0.01	4	3	3	5	0.001	(maxx-minx)/2	0.9642	0.5629	0.8380	0.4380	0.6525
0.0001	4	7	7	7	0.01	(maxx-minx)/2	0.9667	0.5402	0.8540	0.4180	0.6524
0.0001	4	3	3	7	0.01	(maxx-minx)/2	0.9478	0.5585	0.8320	0.4400	0.6515
0.01	2	5	5	7	0.01	(maxx-minx)/2	0.9136	0.5919	0.8200	0.4400	0.6510
0.01	2	5	5	5	0.01	(maxx-minx)/2	0.9058	0.5969	0.8180	0.4440	0.6499
0.0001	4	3	3	5	0.01	(maxx-minx)/2	0.9560	0.5473	0.8420	0.4280	0.6497
0.01	4	7	3	5	0.001	(maxx-minx)/2	0.9632	0.5549	0.8400	0.4320	0.6496
0.0001	4	3	5	10	0.01	(maxx-minx)/2	0.9629	0.5488	0.8460	0.4180	0.6495
0.01	2	7	5	5	0.01	(maxx-minx)/2	0.9177	0.5925	0.8300	0.4280	0.6492
0.01	2	7	5	7	0.01	(maxx-minx)/2	0.9169	0.5803	0.8220	0.4360	0.6473
0.0001	4	5	3	7	0.01	(maxx-minx)/2	0.9608	0.5442	0.8380	0.4280	0.6463
0.01	2	3	3	7	0.01	(maxx-minx)/2	0.8938	0.6162	0.7920	0.4620	0.6449
0.01	2	5	3	7	0.01	(maxx-minx)/2	0.8955	0.6041	0.8100	0.4400	0.6448
0.01	2	3	3	5	0.01	(maxx-minx)/2	0.8886	0.6207	0.7900	0.4660	0.6437
0.01	3	7	7	10	0.01	(maxx-minx)/2	0.9284	0.5524	0.8340	0.4300	0.6425
0.0001	4	7	3	10	0.01	(maxx-minx)/2	0.9563	0.5394	0.8340	0.4240	0.6422
0.01	3	7	7	7	0.01	(maxx-minx)/2	0.9305	0.5607	0.8260	0.4420	0.6422
0.01	3	5	7	7	0.01	(maxx-minx)/2	0.9286	0.5669	0.8200	0.4340	0.6419
0.01	3	3	7	7	0.01	(maxx-minx)/2	0.9149	0.5729	0.8040	0.4600	0.6418
0.01	3	5	7	10	0.01	(maxx-minx)/2	0.9251	0.5561	0.8280	0.4280	0.6388
0.01	2	3	3	10	0.01	(maxx-minx)/2	0.8856	0.6139	0.7880	0.4600	0.6383
0.01	3	3	5	10	0.01	(maxx-minx)/2	0.9116	0.5761	0.8000	0.4580	0.6381
0.0001	4	7	3	7	0.01	(maxx-minx)/2	0.9523	0.5474	0.8140	0.4360	0.6380
0.0001	4	7	5	5	0.01	(maxx-minx)/2	0.9569	0.5472	0.8200	0.4260	0.6357
0.01	2	5	3	10	0.01	(maxx-minx)/2	0.8941	0.5939	0.8020	0.4240	0.6355
0.01	2	7	3	5	0.01	(maxx-minx)/2	0.8845	0.5901	0.8000	0.4420	0.6352
0.01	2	7	3	10	0.01	(maxx-minx)/2	0.8904	0.5873	0.8000	0.4360	0.6345
0.01	2	7	3	7	0.01	(maxx-minx)/2	0.8806	0.5964	0.7920	0.4460	0.6342
0.0001	4	5	3	5	0.01	(maxx-minx)/2	0.9496	0.5412	0.8180	0.4180	0.6305
0.01	3	5	7	5	0.01	(maxx-minx)/2	0.9278	0.5597	0.8080	0.4260	0.6303
0.01	3	7	5	10	0.01	(maxx-minx)/2	0.9076	0.5568	0.8100	0.4340	0.6296
0.0001	4	7	3	5	0.01	(maxx-minx)/2	0.9457	0.5393	0.8120	0.4260	0.6295
0.01	3	7	7	5	0.01	(maxx-minx)/2	0.9223	0.5590	0.8020	0.4380	0.6288
0.01	2	5	3	5	0.01	(maxx-minx)/2	0.8816	0.5958	0.7880	0.4380	0.6287

0.01	3	3	7	5	0.01	(maxx-minx)/2	0.9183	0.5640	0.7940	0.4400	0.6259
0.01	3	3	7	10	0.01	(maxx-minx)/2	0.9185	0.5608	0.7960	0.4380	0.6252
0.01	3	5	5	10	0.01	(maxx-minx)/2	0.9063	0.5594	0.8000	0.4240	0.6222
0.01	3	3	5	5	0.01	(maxx-minx)/2	0.8923	0.5696	0.7700	0.4520	0.6190
0.01	4	7	7	7	0.01	(maxx-minx)/2	0.9112	0.5527	0.7820	0.4420	0.6142
0.01	3	3	5	7	0.01	(maxx-minx)/2	0.8955	0.5675	0.7720	0.4520	0.6140
0.01	3	5	5	7	0.01	(maxx-minx)/2	0.8955	0.5680	0.7680	0.4400	0.6140
0.01	3	5	5	5	0.01	(maxx-minx)/2	0.8896	0.5697	0.7700	0.4460	0.6139
0.01	4	5	7	10	0.01	(maxx-minx)/2	0.9102	0.5549	0.7780	0.4420	0.6138
0.01	3	7	5	7	0.01	(maxx-minx)/2	0.8887	0.5637	0.7800	0.4360	0.6123
0.01	3	3	3	5	0.01	(maxx-minx)/2	0.8716	0.5720	0.7700	0.4480	0.6113
0.01	4	5	7	7	0.01	(maxx-minx)/2	0.9095	0.5583	0.7760	0.4380	0.6080
0.01	4	7	5	10	0.01	(maxx-minx)/2	0.8990	0.5451	0.7820	0.4260	0.6056
0.01	3	5	3	5	0.01	(maxx-minx)/2	0.8705	0.5653	0.7660	0.4360	0.6054
0.01	3	5	3	10	0.01	(maxx-minx)/2	0.8813	0.5565	0.7680	0.4320	0.6053
0.01	3	7	5	5	0.01	(maxx-minx)/2	0.8942	0.5514	0.7760	0.4240	0.6033
0.01	4	7	7	10	0.01	(maxx-minx)/2	0.9051	0.5486	0.7680	0.4300	0.6014
0.01	3	7	3	10	0.01	(maxx-minx)/2	0.8699	0.5520	0.7760	0.4180	0.6012
0.01	3	3	3	10	0.01	(maxx-minx)/2	0.8725	0.5787	0.7480	0.4460	0.6008
0.01	3	7	3	7	0.01	(maxx-minx)/2	0.8730	0.5595	0.7540	0.4380	0.6008
0.01	3	7	3	5	0.01	(maxx-minx)/2	0.8781	0.5626	0.7520	0.4300	0.5996
0.01	4	7	5	7	0.01	(maxx-minx)/2	0.8845	0.5484	0.7520	0.4300	0.5994
0.01	4	5	5	10	0.01	(maxx-minx)/2	0.8839	0.5516	0.7660	0.4220	0.5984
0.01	4	3	7	10	0.01	(maxx-minx)/2	0.8993	0.5550	0.7560	0.4360	0.5980
0.01	3	5	3	7	0.01	(maxx-minx)/2	0.8725	0.5619	0.7580	0.4360	0.5967
0.01	4	3	7	7	0.01	(maxx-minx)/2	0.8976	0.5528	0.7580	0.4320	0.5956
0.01	4	5	7	5	0.01	(maxx-minx)/2	0.9008	0.5476	0.7540	0.4180	0.5936
0.01	4	7	7	5	0.01	(maxx-minx)/2	0.8918	0.5484	0.7580	0.4260	0.5930
0.01	3	3	3	7	0.01	(maxx-minx)/2	0.8708	0.5670	0.7440	0.4320	0.5923
0.01	4	3	5	10	0.01	(maxx-minx)/2	0.8886	0.5493	0.7540	0.4260	0.5911
0.01	4	5	5	5	0.01	(maxx-minx)/2	0.8817	0.5489	0.7480	0.4260	0.5903
0.01	4	3	7	5	0.01	(maxx-minx)/2	0.9017	0.5522	0.7400	0.4360	0.5890
0.01	4	3	5	7	0.01	(maxx-minx)/2	0.8819	0.5553	0.7380	0.4340	0.5890
0.01	4	5	5	7	0.01	(maxx-minx)/2	0.8846	0.5476	0.7520	0.4140	0.5881
0.01	4	7	3	10	0.01	(maxx-minx)/2	0.8603	0.5536	0.7440	0.4300	0.5868
0.01	4	3	3	10	0.01	(maxx-minx)/2	0.8645	0.5624	0.7380	0.4340	0.5861
0.01	4	7	5	5	0.01	(maxx-minx)/2	0.8744	0.5533	0.7320	0.4280	0.5837
0.01	4	3	3	7	0.01	(maxx-minx)/2	0.8571	0.5633	0.7200	0.4460	0.5817
0.01	4	5	3	7	0.01	(maxx-minx)/2	0.8577	0.5507	0.7260	0.4240	0.5755
0.01	4	5	3	10	0.01	(maxx-minx)/2	0.8538	0.5541	0.7260	0.4300	0.5736
0.01	4	3	5	5	0.01	(maxx-minx)/2	0.8719	0.5521	0.7080	0.4380	0.5721
0.01	4	7	3	5	0.01	(maxx-minx)/2	0.8563	0.5534	0.7160	0.4260	0.5702
0.01	4	7	3	7	0.01	(maxx-minx)/2	0.8567	0.5485	0.7180	0.4180	0.5700
0.01	4	5	3	5	0.01	(maxx-minx)/2	0.8617	0.5503	0.7140	0.4200	0.5698
0.01	4	3	3	5	0.01	(maxx-minx)/2	0.8524	0.5587	0.6940	0.4320	0.5610

Table A.2. The detailed results of the experiments with multimodal PSO algorithms when *maxv*=(*maxx*-*minx*)/20

fnc.	Algorithm	osr	gpr	lpr	gscr	lscr	avg. #of fnc. evals
F1	Spec-20	0.9940	0.996	1.000	0.980	1.000	1,113
F1	Niche-20	0.9870	0.988	1.000	0.960	1.000	1,578
F1	nbest-20	0.9040	0.936	1.000	0.680	1.000	6,937
F1	CPSO-20	0.9820	0.988	1.000	0.940	1.000	1,829
F1	Unified-20	0.8600	0.900	1.000	0.540	1.000	7,746
F1	mNiche-20	1.0000	1.000	1.000	1.000	1.000	708
F1	PVPSO-20	1.0000	1.000	1.000	1.000	1.000	933
F2	Spec-20	0.9938	1.000	0.995	1.000	0.980	452
F2	Niche-20	1.0000	1.000	1.000	1.000	1.000	393
F2	nbest-20	0.9325	0.940	0.970	0.940	0.880	1,915
F2	CPSO-20	0.8725	1.000	0.890	1.000	0.600	422
F2	Unified-20	0.8050	1.000	0.840	1.000	0.380	432
F2	mNiche-20	1.0000	1.000	1.000	1.000	1.000	410
F2	PVPSO-20	1.0000	1.000	1.000	1.000	1.000	626
F3	Spec-20	1.0000	1.000	1.000	1.000	1.000	838
F3	Niche-20	0.9700	0.980	1.000	0.900	1.000	2,464
F3	nbest-20	0.8720	0.908	1.000	0.580	1.000	8,953
F3	CPSO-20	0.9520	0.968	1.000	0.840	1.000	3,200
F3	Unified-20	0.8390	0.876	1.000	0.480	1.000	8,293
F3	mNiche-20	1.0000	1.000	1.000	1.000	1.000	738
F3	PVPSO-20	1.0000	1.000	1.000	1.000	1.000	943
F4	Spec-20	0.9413	0.980	0.945	0.980	0.860	892
F4	Niche-20	0.9613	1.000	0.965	1.000	0.880	610
F4	nbest-20	0.8425	0.800	0.950	0.800	0.820	4,864
F4	CPSO-20	0.9000	1.000	0.920	1.000	0.680	473
F4	Unified-20	0.7438	0.940	0.795	0.940	0.300	1,411
F4	mNiche-20	1.0000	1.000	1.000	1.000	1.000	588
F4	PVPSO-20	1.0000	1.000	1.000	1.000	1.000	678
F5	Spec-20	0.9750	0.980	1.000	0.920	1.000	5,979
F5	Niche-20	0.5950	0.360	1.000	0.020	1.000	24,554
F5	nbest-20	0.9488	0.955	1.000	0.840	1.000	9,740
F5	CPSO-20	1.0000	1.000	1.000	1.000	1.000	6,024
F5	Unified-20	0.9375	0.950	1.000	0.800	1.000	8,218
F5	mNiche-20	1.0000	1.000	1.000	1.000	1.000	5,312
F5	PVPSO-20	0.6025	0.410	1.000	0.000	1.000	25,000
F6	Spec-20	0.4781	0.790	0.723	0.400	0.000	16,089
F6	Niche-20	0.2138	0.500	0.255	0.060	0.040	23,662
F6	nbest-20	0.4263	0.770	0.655	0.280	0.000	20,496
F6	CPSO-20	0.6563	0.995	0.650	0.980	0.000	3,120
F6	Unified-20	0.5406	0.960	0.363	0.840	0.000	6,861
F6	mNiche-20	0.6600	0.815	0.945	0.280	0.600	18,842
F6	PVPSO-20	0.3769	0.605	0.723	0.080	0.100	23,492
F7	Spec-20	0.8575	0.910	1.000	0.520	1.000	13,355
F7	Niche-20	0.7225	0.570	1.000	0.320	1.000	17,692

F7	nbest-20	0.8683	0.913	1.000	0.560	1.000	14,859
F7	CPSO-20	1.0000	1.000	1.000	1.000	1.000	2,710
F7	Unified-20	0.7758	0.823	1.000	0.280	1.000	19,116
F7	mNiche-20	0.9942	0.997	1.000	0.980	1.000	3,352
F7	PVPSO-20	0.7275	0.770	1.000	0.140	1.000	23,043
F8	Spec-20	0.2687	0.390	0.525	0.160	0.000	21,878
F8	Niche-20	0.1328	0.500	0.031	0.000	0.000	24,950
F8	nbest-20	0.3122	0.590	0.379	0.280	0.000	21,356
F8	CPSO-20	0.5174	1.000	0.069	1.000	0.000	8,540
F8	Unified-20	0.4792	0.940	0.077	0.900	0.000	5,845
F8	mNiche-20	0.1346	0.000	0.538	0.000	0.000	24,950
F8	PVPSO-20	0.1962	0.180	0.545	0.060	0.000	24,214
F9	Spec-20	0.6356	0.840	0.782	0.840	0.080	6,327
F9	Niche-20	0.6600	1.000	0.620	1.000	0.020	3,000
F9	nbest-20	0.4611	0.600	0.624	0.600	0.020	13,465
F9	CPSO-20	0.6272	1.000	0.509	1.000	0.000	2,266
F9	Unified-20	0.4489	0.700	0.396	0.700	0.000	9,656
F9	mNiche-20	0.4694	0.400	0.878	0.400	0.200	16,782
F9	PVPSO-20	0.3544	0.320	0.738	0.320	0.040	18,766
F10	Spec-20	0.8225	0.990	0.840	0.980	0.480	2,341
F10	Niche-20	0.4513	0.690	0.455	0.380	0.280	16,360
F10	nbest-20	0.5750	1.000	0.280	1.000	0.020	1,917
F10	CPSO-20	0.6150	1.000	0.460	1.000	0.000	1,626
F10	Unified-20	0.5388	0.990	0.185	0.980	0.000	1,621
F10	mNiche-20	0.9088	1.000	0.915	1.000	0.720	1,818
F10	PVPSO-20	0.7050	0.880	0.800	0.780	0.360	10,905

Table A.3. The detailed results of the experiments with multimodal PSO algorithms when *maxv*=(*maxx-minx*)/2

fnc.	**Algorithm**	**osr**	**gpr**	**lpr**	**gscr**	**lscr**	**avg. #of fnc. evals**
F1	Spec-2	0.9880	0.992	1.000	0.960	1.000	2,239
F1	Niche-2	0.7260	0.724	1.000	0.180	1.000	14,970
F1	nbest-2	0.8550	0.900	1.000	0.520	1.000	9,606
F1	CPSO-2	0.9820	0.988	1.000	0.940	1.000	2,455
F1	Unified-2	0.8580	0.892	1.000	0.540	1.000	7,785
F1	mNiche-2	0.8480	0.872	1.000	0.520	1.000	12,196
F1	PVPSO-2	0.9940	0.996	1.000	0.980	1.000	2,353
F2	Spec-2	0.8225	1.000	0.830	1.000	0.460	554
F2	Niche-2	0.6875	1.000	0.750	1.000	0.000	60
F2	nbest-2	0.8538	1.000	0.875	1.000	0.540	625
F2	CPSO-2	0.7813	1.000	0.785	1.000	0.340	443
F2	Unified-2	0.6950	1.000	0.640	1.000	0.140	482
F2	mNiche-2	0.6875	1.000	0.750	1.000	0.000	60
F2	PVPSO-2	0.9250	1.000	0.940	1.000	0.760	773
F3	Spec-2	0.9820	0.988	1.000	0.940	1.000	2,529

F3	Niche-2	0.6980	0.672	1.000	0.120	1.000	14,970
F3	nbest-2	0.8850	0.920	1.000	0.620	1.000	8,096
F3	CPSO-2	0.9150	0.940	1.000	0.720	1.000	5,459
F3	Unified-2	0.8290	0.876	1.000	0.440	1.000	9,803
F3	mNiche-2	0.8040	0.836	1.000	0.380	1.000	12,785
F3	PVPSO-2	0.9880	0.992	1.000	0.960	1.000	2,242
F4	Spec-2	0.8375	1.000	0.830	1.000	0.520	647
F4	Niche-2	0.6388	1.000	0.555	1.000	0.000	729
F4	nbest-2	0.7988	0.960	0.835	0.960	0.440	2,094
F4	CPSO-2	0.7775	1.000	0.790	1.000	0.320	497
F4	Unified-2	0.6950	1.000	0.660	1.000	0.120	578
F4	mNiche-2	0.6663	1.000	0.665	1.000	0.000	656
F4	PVPSO-2	0.9688	1.000	0.975	1.000	0.900	1,163
F5	Spec-2	1.0000	1.000	1.000	1.000	1.000	4,564
F5	Niche-2	0.5625	0.250	1.000	0.000	1.000	24,950
F5	nbest-2	0.9425	0.950	1.000	0.820	1.000	8,865
F5	CPSO-2	0.6388	0.495	1.000	0.060	1.000	24,710
F5	Unified-2	0.9063	0.925	1.000	0.700	1.000	10,689
F5	mNiche-2	1.0000	1.000	1.000	1.000	1.000	4,513
F5	PVPSO-2	0.7225	0.690	1.000	0.200	1.000	22,744
F6	Spec-2	0.6375	0.995	0.555	0.980	0.020	3,657
F6	Niche-2	0.0950	0.380	0.000	0.000	0.000	24,950
F6	nbest-2	0.4713	0.895	0.350	0.640	0.000	12,203
F6	CPSO-2	0.5631	1.000	0.253	1.000	0.000	4,016
F6	Unified-2	0.4738	0.960	0.095	0.840	0.000	5,852
F6	mNiche-2	0.5394	0.925	0.533	0.700	0.000	11,506
F6	PVPSO-2	0.5650	0.905	0.695	0.640	0.020	14,890
F7	Spec-2	0.9825	0.990	1.000	0.940	1.000	4,750
F7	Niche-2	0.5733	0.293	1.000	0.000	1.000	24,950
F7	nbest-2	0.8617	0.907	1.000	0.540	1.000	15,349
F7	CPSO-2	0.9883	0.993	1.000	0.960	1.000	4,498
F7	Unified-2	0.8217	0.887	1.000	0.400	1.000	15,853
F7	mNiche-2	0.9533	0.973	1.000	0.840	1.000	6,445
F7	PVPSO-2	0.8950	0.940	1.000	0.640	1.000	13,755
F8	Spec-2	0.5786	0.990	0.344	0.980	0.000	5,504
F8	Niche-2	0.1301	0.500	0.021	0.000	0.000	24,950
F8	nbest-2	0.4108	0.820	0.183	0.640	0.000	14,523
F8	CPSO-2	0.1726	0.470	0.001	0.220	0.000	23,926
F8	Unified-2	0.5068	1.000	0.027	1.000	0.000	3,116
F8	mNiche-2	0.6192	0.990	0.507	0.980	0.000	6,547
F8	PVPSO-2	0.3930	0.680	0.452	0.440	0.000	19,936
F9	Spec-2	0.6483	1.000	0.593	1.000	0.000	3,218
F9	Niche-2	0.5928	1.000	0.351	1.000	0.020	2,223
F9	nbest-2	0.5194	0.800	0.478	0.800	0.000	9,682
F9	CPSO-2	0.5622	1.000	0.249	1.000	0.000	3,318
F9	Unified-2	0.5639	1.000	0.256	1.000	0.000	1,703
F9	mNiche-2	0.7489	1.000	0.796	1.000	0.200	2,368
F9	PVPSO-2	0.5461	0.740	0.684	0.740	0.020	11,681
F10	Spec-2	0.5925	1.000	0.370	1.000	0.000	2,176
F10	Niche-2	0.1400	0.520	0.000	0.040	0.000	24,950
F10	nbest-2	0.5950	1.000	0.380	1.000	0.000	1,585

F10	CPSO-2	0.5325	1.000	0.130	1.000	0.000	2,098
F10	Unified-2	0.5025	1.000	0.010	1.000	0.000	1,443
F10	mNiche-2	0.6350	1.000	0.540	1.000	0.000	2,216
F10	PVPSO-2	0.6225	1.000	0.470	1.000	0.020	4,059

Table A.4. The detailed results of the experiments on global optimization problems when *maxv*=(*maxx-minx*)/20

fnc.	Algorithm	Dim	Best Fitness		# of succ. runs	avg #of fnc.eval
			avg.	std.dev		
F11	Std.PSO-20	2	0.00003	0.00002	50	61
F11	Std.PSO-20	5	0.32036	1.07737	2	3,901
F11	Std.PSO-20	10	0.49579	1.20919	0	8,000
F11	Std.PSO-20	15	0.59572	0.21684	0	16,000
F11	Std.PSO-20	20	3.58196	1.31681	0	25,000
F12	Std.PSO-20	2	1.24596	1.72179	20	1,229
F13	Std.PSO-20	2	0.00003	0.00001	50	78
F13	Std.PSO-20	5	1.73123	1.02121	4	3,694
F13	Std.PSO-20	10	8.29795	3.63276	0	8,000
F13	Std.PSO-20	15	15.18306	4.52526	0	16,000
F13	Std.PSO-20	20	18.96391	6.28113	0	25,000
F14	Std.PSO-20	2	0.66000	0.55733	20	1,296
F14	Std.PSO-20	5	3.58000	6.37306	3	3,785
F14	Std.PSO-20	10	7.94000	4.72255	2	7,692
F14	Std.PSO-20	15	15.18000	6.34611	0	16,000
F14	Std.PSO-20	20	19.88000	6.85607	0	25,000
F15	Std.PSO-20	2	110.54256	79.66090	13	1,504
F15	Std.PSO-20	5	601.67892	156.36442	0	4,000
F15	Std.PSO-20	10	1686.65539	281.84888	0	8,000
F15	Std.PSO-20	15	2790.67395	519.92535	0	16,000
F15	Std.PSO-20	20	3912.34555	649.43493	0	25,000
F16	Std.PSO-20	2	0.00032	0.00146	48	309
F16	Std.PSO-20	5	0.01487	0.00930	6	3,662
F16	Std.PSO-20	10	0.05288	0.02584	1	7,929
F16	Std.PSO-20	15	0.04587	0.03063	1	15,686
F16	Std.PSO-20	20	0.02213	0.01810	7	21,554
F17	Std.PSO-20	2	0.00004	0.00001	50	128
F17	Std.PSO-20	5	0.00004	0.00001	50	193
F17	Std.PSO-20	10	0.00004	0.00000	50	256
F17	Std.PSO-20	15	0.00005	0.00000	50	319
F17	Std.PSO-20	20	0.00005	0.00000	50	362
F18	Std.PSO-20	2	0.00003	0.00002	50	108
F18	Std.PSO-20	5	0.99983	0.00091	0	4,000
F18	Std.PSO-20	10	1.00000	0.00000	0	8,000
F18	Std.PSO-20	15	1.00000	0.00000	0	16,000
F18	Std.PSO-20	20	1.00000	0.00000	0	25,000
F11	CPSO-20	2	0.00003	0.00002	50	68

F11	CPSO-20	5	0.03296	0.02597	1	1,962
F11	CPSO-20	10	0.33078	0.38358	0	4,000
F11	CPSO-20	15	1.20836	1.30502	0	8,000
F11	CPSO-20	20	1.07374	1.41009	0	12,500
F12	CPSO-20	2	0.01990	0.14057	49	72
F13	CPSO-20	2	0.00003	0.00002	50	64
F13	CPSO-20	5	0.17912	0.43531	42	861
F13	CPSO-20	10	3.80076	1.83890	2	3,903
F13	CPSO-20	15	9.90991	3.16539	0	8,000
F13	CPSO-20	20	10.64614	5.04636	0	12,500
F14	CPSO-20	2	0.00000	0.00000	50	11
F14	CPSO-20	5	0.00000	0.00000	50	73
F14	CPSO-20	10	0.22000	0.50669	41	1,310
F14	CPSO-20	15	0.56000	0.81215	30	4,457
F14	CPSO-20	20	0.12000	0.38545	45	1,908
F15	CPSO-20	2	0.00002	0.00001	50	52
F15	CPSO-20	5	45.00662	58.07201	24	1,660
F15	CPSO-20	10	587.93265	166.02061	0	4,000
F15	CPSO-20	15	1259.99281	199.86564	0	8,000
F15	CPSO-20	20	1994.49411	183.92514	0	12,500
F16	CPSO-20	2	0.00002	0.00002	50	190
F16	CPSO-20	5	0.01344	0.00642	0	2,000
F16	CPSO-20	10	0.02130	0.01419	0	4,000
F16	CPSO-20	15	0.00626	0.00791	0	8,000
F16	CPSO-20	20	0.00029	0.00006	0	12,500
F17	CPSO-20	2	0.00004	0.00001	44	537
F17	CPSO-20	5	0.00058	0.00014	0	2,000
F17	CPSO-20	10	0.00166	0.00018	0	4,000
F17	CPSO-20	15	1.08421	1.27106	0	8,000
F17	CPSO-20	20	0.31435	0.64937	0	12,500
F18	CPSO-20	2	0.00002	0.00001	50	120
F18	CPSO-20	5	1.00000	0.00000	0	2,000
F18	CPSO-20	10	1.00000	0.00000	0	4,000
F18	CPSO-20	15	1.00000	0.00000	0	8,000
F18	CPSO-20	20	1.00000	0.00000	0	12,500
F11	Uni.PSO-20	2	0.00003	0.00002	50	95
F11	Uni.PSO-20	5	0.00040	0.00018	0	4,000
F11	Uni.PSO-20	10	0.00040	0.00024	0	8,000
F11	Uni.PSO-20	15	0.00005	0.00000	50	13,645
F11	Uni.PSO-20	20	0.00005	0.00000	50	17,513
F12	Uni.PSO-20	2	0.15827	0.72878	46	296
F13	Uni.PSO-20	2	0.00003	0.00002	50	57
F13	Uni.PSO-20	5	0.07963	0.27266	46	1,427
F13	Uni.PSO-20	10	4.76558	1.82829	0	8,000
F13	Uni.PSO-20	15	11.96741	3.09566	0	16,000
F13	Uni.PSO-20	20	19.77610	4.24854	0	25,000
F14	Uni.PSO-20	2	1.54000	1.63145	10	1,609
F14	Uni.PSO-20	5	2.50000	1.86537	6	3,597
F14	Uni.PSO-20	10	0.96000	1.08722	19	5,520
F14	Uni.PSO-20	15	1.34000	1.28746	18	10,881
F14	Uni.PSO-20	20	0.84000	0.97646	22	15,768

F15	Uni.PSO-20	2	25.66172	53.54232	40	538
F15	Uni.PSO-20	5	408.60501	151.46357	2	3,885
F15	Uni.PSO-20	10	1193.00799	221.77732	0	8,000
F15	Uni.PSO-20	15	2050.38039	286.43357	0	16,000
F15	Uni.PSO-20	20	2972.49288	321.43313	0	25,000
F16	Uni.PSO-20	2	0.00003	0.00002	50	207
F16	Uni.PSO-20	5	0.01796	0.01169	4	3,847
F16	Uni.PSO-20	10	0.02472	0.01671	4	7,449
F16	Uni.PSO-20	15	0.00206	0.00377	38	4,450
F16	Uni.PSO-20	20	0.00005	0.00000	50	598
F17	Uni.PSO-20	2	0.00003	0.00001	50	69
F17	Uni.PSO-20	5	0.00004	0.00001	50	145
F17	Uni.PSO-20	10	0.00005	0.00000	50	276
F17	Uni.PSO-20	15	0.00005	0.00000	50	409
F17	Uni.PSO-20	20	0.00005	0.00000	50	524
F18	Uni.PSO-20	2	0.91089	0.27884	3	1,904
F18	Uni.PSO-20	5	1.00000	0.00000	0	4,000
F18	Uni.PSO-20	10	1.00000	0.00000	0	8,000
F18	Uni.PSO-20	15	1.00000	0.00000	0	16,000
F18	Uni.PSO-20	20	1.00000	0.00000	0	25,000

Table A.5. The detailed results of the experiments on global optimization problems when *maxv*=(*maxx-minx*)/2

fnc.	Algorithm	Dim	Best Fitness		# of succ. runs	avg #of fnc. eval
			avg.	std.dev		
F11	Std.PSO-2	2	0.00002	0.00001	50	106
F11	Std.PSO-2	5	0.10939	0.55289	1	3,924
F11	Std.PSO-2	10	1.87615	0.85670	0	8,000
F11	Std.PSO-2	15	6.45115	1.47493	0	16,000
F11	Std.PSO-2	20	11.08133	2.49229	0	25,000
F12	Std.PSO-2	2	0.00001	0.00001	50	68
F13	Std.PSO-2	2	0.00002	0.00001	50	106
F13	Std.PSO-2	5	0.11943	0.32659	44	884
F13	Std.PSO-2	10	2.08942	1.12369	4	7,465
F13	Std.PSO-2	15	5.87026	2.92159	2	15,676
F13	Std.PSO-2	20	8.37755	6.18937	1	24,748
F14	Std.PSO-2	2	0.00000	0.00000	50	124
F14	Std.PSO-2	5	0.00000	0.00000	50	268
F14	Std.PSO-2	10	0.00000	0.00000	50	330
F14	Std.PSO-2	15	0.00000	0.00000	50	429
F14	Std.PSO-2	20	0.00000	0.00000	50	464
F15	Std.PSO-2	2	4.73756	23.44470	48	200
F15	Std.PSO-2	5	113.70084	117.12582	21	2,417
F15	Std.PSO-2	10	315.04607	170.22660	0	8,000
F15	Std.PSO-2	15	556.66034	218.32395	0	16,000
F15	Std.PSO-2	20	783.66727	242.06523	0	25,000

F16	Std.PSO-2	2	0.00034	0.00146	47	476
F16	Std.PSO-2	5	0.01804	0.01163	6	3,673
F16	Std.PSO-2	10	0.05612	0.02453	1	7,882
F16	Std.PSO-2	15	0.03711	0.02868	2	15,586
F16	Std.PSO-2	20	0.02156	0.01809	11	19,594
F17	Std.PSO-2	2	0.00003	0.00001	50	145
F17	Std.PSO-2	5	0.00004	0.00001	50	215
F17	Std.PSO-2	10	0.00005	0.00000	50	284
F17	Std.PSO-2	15	0.00005	0.00000	50	355
F17	Std.PSO-2	20	0.00005	0.00000	50	408
F18	Std.PSO-2	2	0.00003	0.00002	50	121
F18	Std.PSO-2	5	1.00000	0.00000	0	4,000
F18	Std.PSO-2	10	1.00000	0.00000	0	8,000
F18	Std.PSO-2	15	1.00000	0.00000	0	16,000
F18	Std.PSO-2	20	1.00000	0.00000	0	25,000
F11	CPSO-2	2	0.00004	0.00002	50	70
F11	CPSO-2	5	0.04754	0.03546	1	2,048
F11	CPSO-2	10	0.40160	0.44003	0	4,000
F11	CPSO-2	15	1.42389	1.57813	0	8,000
F11	CPSO-2	20	1.44341	1.98795	0	12,500
F12	CPSO-2	2	0.02799	0.14394	49	74
F13	CPSO-2	2	0.00003	0.00002	50	70
F13	CPSO-2	5	0.26452	0.58444	42	940
F13	CPSO-2	10	5.26580	1.99734	2	4,043
F13	CPSO-2	15	9.90991	3.22735	0	8,000
F13	CPSO-2	20	10.87613	5.43718	0	12,500
F14	CPSO-2	2	0.00000	0.00000	50	11
F14	CPSO-2	5	0.00000	0.00000	50	80
F14	CPSO-2	10	0.23038	0.53531	41	1,387
F14	CPSO-2	15	0.57146	0.88732	30	4,468
F14	CPSO-2	20	0.12707	0.42218	45	1,963
F15	CPSO-2	2	0.00002	0.00001	50	53
F15	CPSO-2	5	46.43346	60.27287	24	1,689
F15	CPSO-2	10	636.67743	178.34239	0	4,000
F15	CPSO-2	15	1335.45209	202.61709	0	8,000
F15	CPSO-2	20	2150.45721	200.45226	0	12,500
F16	CPSO-2	2	0.00002	0.00002	50	200
F16	CPSO-2	5	0.01393	0.00677	0	2,000
F16	CPSO-2	10	0.02219	0.01516	0	4,000
F16	CPSO-2	15	0.00648	0.00807	0	8,000
F16	CPSO-2	20	0.00029	0.00006	0	12,500
F17	CPSO-2	2	0.00004	0.00001	44	557
F17	CPSO-2	5	0.00081	0.00015	0	2,000
F17	CPSO-2	10	0.00205	0.00022	0	4,000
F17	CPSO-2	15	1.23652	1.74136	0	8,000
F17	CPSO-2	20	0.45919	0.67303	0	12,500
F18	CPSO-2	2	0.00003	0.00002	50	127
F18	CPSO-2	5	1.00000	0.00000	0	2,000
F18	CPSO-2	10	1.00000	0.00000	0	4,000
F18	CPSO-2	15	1.00000	0.00000	0	8,000
F18	CPSO-2	20	1.00000	0.00000	0	12,500

F11	Uni.PSO-2	2	0.00003	0.00002	50	105
F11	Uni.PSO-2	5	0.00041	0.00021	2	3,884
F11	Uni.PSO-2	10	0.00072	0.00068	0	8,000
F11	Uni.PSO-2	15	0.00005	0.00001	44	14,669
F11	Uni.PSO-2	20	0.00005	0.00000	50	18,484
F12	Uni.PSO-2	2	0.00002	0.00002	50	102
F13	Uni.PSO-2	2	0.00003	0.00001	50	59
F13	Uni.PSO-2	5	0.01994	0.14070	49	780
F13	Uni.PSO-2	10	3.82520	2.24012	2	7,803
F13	Uni.PSO-2	15	14.83422	4.16393	0	16,000
F13	Uni.PSO-2	20	28.17024	7.37040	0	25,000
F14	Uni.PSO-2	2	0.00000	0.00000	50	7
F14	Uni.PSO-2	5	0.38000	0.49031	31	1,774
F14	Uni.PSO-2	10	1.70000	0.93132	4	7,453
F14	Uni.PSO-2	15	4.18000	1.25666	0	16,000
F14	Uni.PSO-2	20	6.68000	1.46301	0	25,000
F15	Uni.PSO-2	2	0.00002	0.00002	50	50
F15	Uni.PSO-2	5	0.00005	0.00000	50	360
F15	Uni.PSO-2	10	112.53630	130.68326	0	8,000
F15	Uni.PSO-2	15	728.01662	343.36601	0	16,000
F15	Uni.PSO-2	20	1449.75367	368.90836	0	25,000
F16	Uni.PSO-2	2	0.00003	0.00002	50	255
F16	Uni.PSO-2	5	0.01614	0.00994	4	3,883
F16	Uni.PSO-2	10	0.02180	0.01506	3	7,553
F16	Uni.PSO-2	15	0.00167	0.00475	42	3,661
F16	Uni.PSO-2	20	0.00024	0.00139	49	1,431
F17	Uni.PSO-2	2	0.00003	0.00001	50	74
F17	Uni.PSO-2	5	0.00004	0.00001	50	153
F17	Uni.PSO-2	10	0.00005	0.00000	50	290
F17	Uni.PSO-2	15	0.00005	0.00000	50	425
F17	Uni.PSO-2	20	0.00005	0.00000	50	554
F18	Uni.PSO-2	2	0.40002	0.49486	30	871
F18	Uni.PSO-2	5	1.00000	0.00000	0	4,000
F18	Uni.PSO-2	10	1.00000	0.00000	0	8,000
F18	Uni.PSO-2	15	1.00000	0.00000	0	16,000
F18	Uni.PSO-2	20	1.00000	0.00000	0	25,000

REFERENCES

1. Wachowiak, M.P. and R. Smolíková, Y. Zheng, J.M. Zurada, A.S. Elmaghraby, *"An Ap-proach to Multimodal Biomedical Image Registration Utilizing Particle Swarm Optimization"*, IEEE Transactions on Evolutionary Computation, vol. 8, no. 3, pp. 289-301, 2004.
2. Cockshott, A.R. and B.E. Hartman, *"Improving the fermentation medium for echinocandin B production Part II: Particle swarm optimization"*, Process Biochem., v. 36, pp. 661–669, 2001.
3. Abido, M.A., *"Optimal power flow using particle swarm optimization"*, Elec. Power, Energy Syst., vol. 2, pp. 563–571, 2002.
4. Thomsen, R., *"Multimodal Optimization Using Crowding-Based Differential Evolution"*, In Proceedings of the 2004 Congress on Evolutionary Computation, vol. 2, pp. 1382-1389, 2000.
5. Back, T. and D.B. Fogel, T. Michalewicz (editors), *Advanced Algorithms and Operators, volume 2 of Evolutionary Computation*, Institute of Phsysics Publishing, Bristol and Philidelphia, 1999.
6. Back, T. and D.B. Fogel, T. Michalewicz (editors), *Basic Algorithms and Operators, volume 1 of Evolutionary Computation*, Institute of Phsysics Publishing, Bristol and Philidelphia, 1999.
7. Jian-Ping Li and M.E Balazs, G.T. Parks, P.J. Clarkson, *"A Genetic Algorithm using Species Conservation for Multimodal Function Optimization"*, Journal of Evolutionary Computation, Volume 10, Number 3, pp. 207-234, 2002.
8. Glover, F., *Tabu Search – Part I*, ORSA Journal on Computing, 1(3):190–206, 1989.
9. Beasley, D. and D.R. Bull, R.R. Martin, *"A Sequential Niche Technique for Multimodal Function Optimization",* Evolutionary Computation, 1(2), pp. 101–125, 1993.
10. Eiben, A.E. and J.E. Smith, *Introduction to Evolutionary Computing*, Springer, ISBN 3-540-40184-9, 2003.
11. Cohoon, J.P. and S.U. Hedge, W.N. Martin, D.S. Richards, *"Punctuated equilibria: a parallel genetic algorithm"*, Proceedings of the Second International Conference on

Genetic Algorithms (ICGA'87), J.J. Grefenstette, ed. Cambridge, MA, USA, pp. 148-154, 1987.

12. Collins, R.J. and D.R. Jefferson, *"Selection in massively parallel genetic algorithms"*, Proceedings of the Fourth International Conference on Genetic Algorithms (ICGA'91), R.K. Belew and L.B.Brooker, eds., San Diego, CA, USA, pp. 249-256, 1991.
13. Tulai, A.F. and F. Oppacher, *"Maintaining diversity and increasing the accuracy of classification rules through automatic speciation"*, Evolutionary Computation, CEC2004, Congress on, 19-23 June 2004,Volume: 2, page(s): pp. 2241- 2249, 2004.
14. Brits, R., *Niching Strategies for Particle Swarm Optimization, MSc. Thesis*, Faculty of Natural & Agricultural Science, University of Pretoria, 2002.
15. Holland, J. H., *Adaptation in Natural and Artificial Systems*, Ann Arbor, MI, Univ. of Michigan Press, 1975.
16. Goldberg, D. E. and J. Richardson, *"Genetic algorithms with sharing for multimodal function optimization"*, in Proc. 2nd Int. Conf. Genetic Algorithms, J. J. Grefensette, Ed. Hillsdale, NJ: Lawrence Erlbaum, pp. 41–49, 1987.
17. Parsopoulos, K. E. and M.N. Vrahatis, *"Modification of the particle swarm optimizer for locating all the global minima"*, Proceedings of the International Conference on Artificial Neural Networks and Genetic Algorithms (ICANNGA 2001), pp. 324-327, 2001.
18. DeJong, K. A., *"An analysis of the behavior of a class of genetic adaptative systems"*, Ph.D. dissertation, Univ. of Michigan, Ann Arbor, 1975.
19. Mahfoud, S.W., *Niching methods for genetic algorithm*, Ph.D. dissertation, Univ. Of Illinois, Urbana-Champaign, 1995.
20. Harik, G., *"Finding multimodal solutions using restricted tournament selection"*, in Proc. 6th Int. Conf. Genetic Algorithms, L. J. Eshelman, Ed. San Mateo, CA: Morgan Kaufmann, pp. 24–31, 1995.
21. P´etrowski, A., *"A clearing procedure as a niching method for genetic algorithms"*, in Proc. 1996 IEEE Int. Conf. Evolutionary Computation, Nagoya, Japan, pp. 798–803, 1996.
22. G. Flake, *The Computational Beauty of Nature*, Cambridge, MA, MIT Press, 1999.
23. Kennedy, J. and R.C. Eberhart, *"Particle Swarm Optimization"*, Proc. IEEE Int. Conf. on N.N., Piscataway,NJ, pp. 1942-1948, 1995.

24. Kennedy., J., *"The particle swarm: social adaptation of knowledge"*, Proc. of the 1997 IEEE Int. Conf. on Evolutionary Computation (Indianapolis, Indiana), pp. 303-308. IEEE Service Center, Piscataway, NJ. 1997.
25. Ozcan, E. and C.K. Mohan, *"Particle Swarm Optimization: Surfing the Waves"*, Proc. of IEEE Congress on Evolutionary Computation, Piscataway, NJ. pp. 1939-1944, 1999.
26. van den Bergh, F., *An Analysis of Particle Swarm Optimizers*, PhD Thesis, Department of Computer Science, University of Pretoria, 2002.
27. Engelbrecht, A.P.,*Computational Intelligence*, An Introduction, Wiley, 2002.
28. Carlisle, A. and G. Dozier, *"An Off-The-Shelf PSO"*, In Proceedings of the Workshop on PSO, Indianapolis, IN, USA, 2001.
29. Franken, N. and A.P. Engelbrecht, *"PSO approaches to co-evolve IPD strategies"*, in Proc. Congress on Evolutionary Computation Portland, OR, vol. 1, pp. 356-363, 2004.
30. van den Bergh, F. and A.P. Englebrecht, *"A study of particle swarm optimization particle trajectories"*, Information Sciences 176, pp. 937–971, 2006.
31. Ozcan, E. and C.K. Mohan, *"Analysis of a simple particle swarm optimization system"*, In Intelligent Engineering Systems Through Artificial Neural Networks, pp. 253–258, Oct. 1998.
32. van den Bergh, F. and A.P. Engelbrecht, *"A New Locally Convergent Particle Swarm Optimiser"*, Conference on Systems, Man and Cybernetics. pp. 96-101, 2002.
33. Peer, E.S. and F. van den Bergh, A.P. Engelbrecht, *"Using Neighborhoods with the Guaranteed Convergence PSO"*, In Proceedings of the IEEE. Swarm Intelligence Symposium, USA, pp. 235-242, 2003.
34. Li, X., *"Adaptively Choosing Neighborhood Bests using Species in a Particle Swarm Optimizer for Multimodal Function Optimization",* In Proceeding of Genetic and Evolutionary Computation Conference 2004 (GECCO'04), Lecture Notes in Computer Science (LNCS), Vol. 3102, pp. 105-116, Springer-Verlag, 2004.
35. Brits, R. and A.P. Engelbrecht, F. van den Bergh, *"A niching particle swarm optimizer"*, Proc. 4-th Asia-Pacific Conf. on Simulated Evolution and Learning, 2002.

36. Løvbjerg, M. and T.K. Rasmussen, T. Krink, *"Hybrid Particle Swarm Optimizer with Breeding and Subpopulations",* In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO), San Francisco, 2001.
37. Thiémard, E., *Economic Generation of Low-Discrepancy Sequences with a b-ary Gray Code*, Technical Report RO981201, Department of Mathematics, Ecole Polytechnique Federale de Lausanne, CH-1015 Lausanne, Switzerland, 1998.
38. Parsopoulos, K. E. and V.P. Plagianakos, G.D. Magoulas, M.N. Vrahatis, *"Improving the Particle Swarm Optimizer by Function 'Stretching' "*. Nonconvex Optimization and its Applications, Kluwer Academic Publishers, VOL 54, pp. 445-458, 2001.
39. Parsopoulos, K.E. and M.N. Vrahatis, *UPSO: A Unified Particle Swarm Optimization Scheme, Lecture Series on Computer and Computational Sciences, Vol. 1*, Proceedings of the International Conference of Computational Methods in Sciences and Engineering (ICCMSE 2004), pp. 868-873, VSP International Science Publishers, Zeist, The Netherlands, 2004.
40. Parsopoulos, K.E. and M.N. Vrahatis, *"Unified Particle Swarm Optimization for Tackling Operations Research Problems"*, IEEE 2005 Swarm Intelligence Symposium (SIS 2005), Pasadena (CA), USA, pp. 53-59, 2005.
41. Eberhart, R. C. and Y. Shi, *"Comparing inertia weights and constriction factors in particle swarm optimization"*, Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2000), San Diego, CA. pp. 84-88, 2000.
42. Schoeman, I.L. and A.P Engelbrecht, *"A Parallel Vector-Based Particle Swarm Optimizer"*, In Proceedings of the International Conference on Neural Networks and Genetic Algorithms, pages 268--271, 2005.
43. Schoeman, I.L. and A.P. Engelbrecht, *"Using vector operations to identify niches for particle swarm optimization"*, Proceedings of the conference on cybernetics and intelligent systems, 2004.
44. Owechko, Y. and S. Medasani, N. Srinivasa, *"Classifier Swarms for Human Detection in Infrared Imagery"*, 2004 Conference on Computer Vision and Pattern Recognition Workshop (CVPRW'04) Volume 8, pp. 121, 2004.
45. Streichert, F. and G. Stein, H. Ulmer, A. Zell, *A Clustering Based Niching EA for Multimodal Search Spaces*, Artificial Evolution, pp. 293-304, 2003.

46. Ursem, R.K., *"Multinational evolutionary algorithms"*, In Proceedings of the 1999 Congress of Evolutionary Computation (CEC-1999), vol. 3, pp. 1633–1640, 1999.
47. Sotiropoulos, D.G. and V.P. Plagianakos, M.N. Vrahatis, *"An evolutionary algorithm for minimizing multimodal functions"*, in proc. of the Fifth Hellenic-European Conference on Computer Mathematics and its Applications (HERCMA 2001), vol. 2, pp. 496-500, 2002.
48. Schwefel, H. P., *Numerical Optimization of Computer Models*, John Wiley & Sons, 1981, English translation of Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie, 1977.
49. Griewangk, A.O., *Generalized Descent of Global Optimization*, Journal of Optimization Theory and Applications, 34: 11.39, 1981.
50. Ackley, D., *An Empirical Study of Bit Vector Function Optimization*, Genetic Algorithms and Simulated Annealing, pp. 170-215, 1987.
51. Brits, R. and A.P. Engelbrecht, F. van den Bergh, *"Solving Systems of Unconstrained Equations using Particle Swarm Optimization",* IEEE International Conference on Systems, Man and Cybernetics, Hammamet, Tunisia, 2002.
52. Wolpert, D.H. and W.G. Macready, *No Free Lunch Theorems for Search*, Technical Report SFI-TR-95-02-010 (Santa Fe Institute), 1995.