International Journal of Computational and Experimental Science and ENgineering (IJCESEN)
Vol. 6-No.1 (2020) pp. 42-62
http://dergipark.org.tr/en/pub/ijcesen

ISSN: 2149-9144

Research Article

Evolutionary Search Algorithm Based on Hypercube Optimization For High-Dimensional Functions

Mustafa TUNAY*

Istanbul Gelisim University, Faculty of Engineering and Architecture, Computer Engineering, Istanbul-Turkey

* **Corresponding Author** : mustafatunay44@hotmail.com
ORCID: 0000-0001-8843-621X

Article Info:

DOI: 10.22399/ijcesen.684653
Received : 4 February 2020
Accepted : 16 March 2020

Keywords

Evolutionary algorithms
Optimization algorithms
Benchmark function
Timetabling problem

Abstract:
This study paper is devoted to the design of a novel evolutionary search algorithm to solve optimization of multivariate systems. Nowadays for many real world optimization problems the designing of evolutionary search algorithm with high optimization accuracy is a crucial optimization of multivariate systems. The proposed optimization search algorithm is a new intense stochastic search method that is based on a hypercube evolution driven. This algorithm is inspired from the behaviour of a pigeon that discovers new location of areas for seeds in natural world. The hypercube is used a statement that shows the area of life for the behaviour of a pigeon in real life.
The performance of the proposed algorithm is tested on optimization functions as some Benchmark function and test suite functions including; four unimodal functions and composition functions. The performance of the proposed algorithm are shown much better experimental results in EAs and is encouraging in terms of solution accuracy for global optimization. In addition, the proposed algorithm approach is applied to solve a timetabling problem as two exams in adjoining periods, conflict of exams, two or more exams in one day etc. They are very difficult to solve for many institutions of higher education and as resulted in a significant increase in their complexity.

1. Introduction

In this study review of optimization search algorithms are given. A derivative based and derivative free optimization search algorithms are specified. The importance of evolutionary search algorithms for multimodal optimization is shown. The research is based on evolutionary search algorithms including genetic algorithms, differential evaluation, particle swarm optimization, ant colony optimization and other evolutionary search algorithms have been analysed. The design of the high accuracy evolutionary search algorithm for solving complex multidimensional optimization problems is noticed.
In many real-world optimization problems, different optimization strategies have been designed for finding solution of these kinds of issues. One of them is the simultaneous perturbation stochastic approximation (SPSA) method for multivariate optimization issues. For this purpose in the proposed method has been developed considerable application in areas such as statistical parameter estimation, feedback control, signal and image processing, simulation-based optimization and experimental design. In problem solution the proposed method uses gradient approximation in any case of dimension of the optimization problem. The SPSA method decreases especially in problems to be optimized for cost of optimization solutions. For more details are referred in [1].
Many optimization problems mainly consist in finding the "best solution" from the values of objective function within certain ranges. The solutions of nonlinear optimization acquire great

importance. These optimization problems can have multiple local optimal (minimum and maximum) solutions. The basic problem is to find the best of these local optimal solutions. The aim of global optimization is to find a feasible region solutions x in a solution space set X, for which the objective function F obtains its smallest (or largest) value.

One of widely used adaptive heuristic search algorithm used for multiobjective optimization. The proposed algorithm is relies on the evolutionary conception of natural selection. By means of Genetic algorithms (GAs) is inspired by natural evolution using searching problems techniques. The Genetic Algorithms (GAs) mainly consist of selection process, a crossover process and a mutation process for search optimization solutions [10]. The natural evolution is generated individuals from a population randomly. The population is selected according to fitness values. The best population is more likely to be selected. Thus, the populations have been selected based on their physical condition and so we excepted that selected population is among the strongest in the population and so we excepted that selected population will gradually increase in the average fitness, used in the next iteration, as the current solution [2].

Genetic algorithms provides in complex adaptive systems for in economic theory to the use of machine learning methods. Adaptation is a biological process survives in environments confronting organisms that evolve by rearranging genetic material. Holland presents using mathematical model that seek out a solution for nonlinearity such as complex interactions [25].

The modification of genetic algorithm is a biased random key using for solving tactical berth allocation problem (TBAP) aiming. The TBAP aiming allocates incoming ships to berthing positions for assigning quay crane profiles (i.e. number of quay cranes per time step) [14].

The design of the TBAP are both the minimization of the housekeeping expenses; first one got from the transhipment compartment streams in the middle of ships, second one is the amplification of the aggregate estimation of the quay crane profiles doled out of the ships. The acquired results for handling the TBAP have demonstrated that the proposed calculation which is appropriate to proficiently take care of this issue.

A new structured population approach is built a hierarchy of hypercube is represented as population of GA. GA is about structured population that generally leads to higher performance than the palmitic genetic algorithm; because it can control two opposite processes, namely exploration and exploitation in the search space. GA is about several spatially structured populations that were introduced in the literature [9, 19]. These are cellular, GA-social patchwork basis, island-style or model, terrain-based, graph-based and religion-based. This research does not build subpopulations based on the information of the genes of individuals themselves. The structuring of subpopulations could help to achieve better performance and more efficient search strategy. The algorithm can dynamically build the structure of a population by dividing the search space. The structured population is represented as hierarchical hypercube subpopulations that are dynamically built and adapted to the search time. Each subpopulation represents a subdivision of the real space of genes. This structure could help guide research towards the promising sub-areas.

A new tendency search optimization is modified artificial chromosomes with genetic algorithm (ACGA). The proposed algorithm has been applied real world problems successfully in order to solve scheduling problems. However, ACGA cannot perform function well in some planning problems due to the fact that its probabilistic model does not take into account the variable interactions, in particular if the sequence-dependent setup times are taken into account. This is due to the fact that the previous job will improve variable interactions for influence the processing time. The improvement of artificial chromosomes with genetic algorithm (ACGA) is successfully applied single machine scheduling problems. This improvement of ACGA is a bi-variate probabilistic model called an ACGA II. The design of ACGA II has very broad concept including some heuristics and local search algorithm, variable neighbourhood search (VNS) [11]. The proposed is successfully demonstrated solving single machine scheduling problems with sequence-dependent setup times for date environment.

Traditional computational intelligent systems are basically based on private "internal" cognitive and

computational processes. However, Swarm Intelligence argues that human intelligence comes from the interactions of individuals in a social world. This proposed model of intelligence commonly can be used in artificial intelligent systems.

The foundations of approach presents through social psychology, cognitive science, and evolutionary computation. The authors describe is referred in [4,8]. The Particle swarm optimization (PSO) algorithm is evaluated objective function in a search space. Each particle determines its movement by using the history of its own current and best locations with the member of swarm and with the random perturbations. The swarm like a flock of birds foraging for a food and move close to optimum point. The proposed algorithm provides a problem solving method.

There are in many real-world optimization problems to use mathematical algorithms that seek an iterative solution because the function or the constraints of objective problem can be modified over time. If these cases are undefined past in the optimization process, we are called dynamic for these types of problems. There are some difficulties in optimizing dynamic environments with the goal that the calculations for rationalization in these situations would be to use some systems keeping in mind the ultimate objective of overcoming difficulties. There are many algorithms for optimization problems. One of them is a new optimization algorithm based on dynamic environments the particle swarm optimization (PSO) in which a new mechanism has been carried out for improving solutions in [23]. In this mechanism, it is attempted to increase local research capacity around with optimal focusing on the best pic found in each environment. Experimental and comparative results demonstrated the superiority of the proposed method.

The particle swarm optimization (PSO) is modified a new cooperative coevolving with PSO (CCPSO) algorithm to optimize large-scale and complex multimodal optimization problems. The updating rules of the PSO is also changed a new cooperative coevolving with PSO (CCPSO II) algorithm. The new proposed algorithm is based on Cauchy and Gaussian method distributions to sample new points in the search space. The design of CCPSO II scheme dynamically determines the coevolving subcomponent sizes of the variables. For more description of details are referred in [26].

The performance of CCPSO II was tested on large-scale and complex multimodal optimization problems. The experimental results have demonstrated the performance of CCPSO II that is successfully applied solving many difficult optimization search problems. The performance of CCPSO II is successfully also evaluated by considering the cases in which the problem dimension are as a set of high dimensional problems.

There is a new alternative way for PSO to optimize multimodal problems. The PSO is changed and modified creating a new design of comprehensive learning particle swarm optimizer (CLPSO). The performance of CLPSO was successfully tested on multimodal problems. The design of CLPSO is applied in a novel learning strategy. This strategy allows the diversity of the swarm to be preserved to discourage premature convergence so that the updating of the appropriate speed for PSO provides best information. For details are briefly referred in [27].

The particle swarm optimization builds the differing qualities of the particles such as a two-layer is proposed. The downside of catching in a neighbourhood ideal is kept away from a structure with two layers (upper layer and lower layer) is proposed [20]. Swarms of particles and one swarm of particles are created in the lower layer and the upper layer respectively. Each better global position in every swarm of the lower layer is to be the position of the particle in the swarm of the upper layer. A variety of particles increases to avoid capture in a local optimum. A mutation is also included into the particles of every swarm in the lower layer so that the particles jump the neighbourhood ideal to locate the global optimum. The proposed algorithm was tested on various types of large-scale optimization problems and experimental results have shown convergence properties successfully. The proposed algorithm works much better for types of large-scale optimization problems.

Nowadays there are many the popular trends to optimize power flow. The well-known is both popular algorithms, GAs and PSO are applied to optimize the power flow. A direction particle swarm optimization algorithm of ant evolution to solve the optimal power flow problem with non-smooth and

non-convex cost characteristics of the generator is considered. Here, the search for ant colony is used to find an update operator of the appropriate speed for particle swarm optimization and ant colony emerge parameter settings using genetic algorithm approach. The updating of the appropriate speed for particle swarm optimization has five operators used.

The proposed method is tested on mainly including as 30-bus IEEE, 39-bus IEEE and 57-bus IEEE - systems with three different objective functions [15]. The simulation results have demonstrated better results for the proposed method. The simulation results have demonstrated the proposed optimization search algorithm that gives better results and it is comparable with classical particle swarm optimization for solving a set of optimization problems.

The swarm-inspired projection (SIP) is inspired by data projection algorithm for swarm optimization. The algorithm makes it possible to visually estimate the number of existing clusters in a data set in [18]. The results are based on the projection then we can partition the data to put in the corresponding number of clusters.

The design of simulated-annealing (SA) algorithm is employed to design fuzzy control systems. SA algorithms are provided for minimizing the objective functions. A design of PI-FCs (Takagi-Sugeno proportional-integral fuzzy controllers) is proposed using SA. The proposed algorithm is implemented on a case study on the control of the angular position of a servo system. The system is used as test bed to confirm the controller design in [12, 21].

There is a new differential evolution (DE) algorithm to develop about the performance of a new mutation strategy. This is a called 'DE/current-to-p-best' which is about the performance of a new mutation strategy. This mutation strategy is a generalization of 'DE/current-to-best' then. This is a called JADE. It differentiates the population. However it still gets the fast convergence property so that self-adaptation is used to improve for its performance. For more details are referred in [28]. In this study avoids the requirement for parameter settings and so that it works well without user interaction.

Self-adaptive control parameter settings are inspired by the DE (differential evolution) algorithm. The differential evolution has been applied practical situation successfully in order to solve many difficult optimization search problems. The proposed algorithm has demonstrated the best convergence features. The proposed algorithm is based on only a few control parameters, which are kept constant throughout the whole evolutionary process, but it is difficult to properly arrange control parameters in DE [24, 30].

Self-adaptation of the mutation distribution in evolutionary strategies develops two useful methods. These are the ideas of de-randomization and cumulating. Standard deficiencies of the idea of mutative strategy parameter control and two levels of de-randomization are looked into. Fundamental requests on the self-adjustment of self-assertive (ordinary) transformation circulations are produced in [17]. Applying subjective, typical mutation distributions is equal to applying a general, straight issue encoding. The performances of these schemes are comparable only on perfectly scaled functions. On severely scaled, non-distinguishable functions for the most part a velocity up variable of a few requests of greatness is watched. On decently mis-scaled functions a velocity up factor of three to ten can be expected.

There is a new heuristic approach is applied for minimizing nonlinear and non-differentiable continuous space functions in [5]. The proposed algorithm selects the difference vector of two randomly. By means of the proposed algorithm perturbs an existing vector for chosen population vectors. The perturbation is done for every population vector. The proposed method is demonstrated converges faster for multi objective optimization.

There is also a viable new approach to stochastic combinatorial optimization is inspired by the behaviour of ant. The proposed algorithm mainly features are constructive greedy heuristic, distributed of computation and positive of feedback. Firstly the greedy heuristic finds acceptable solutions for the search process. Secondly the distributed of computation avoids premature convergence. Finally the positive of feedback explains fast discovery of the best solutions. The proposed methodology is applied in practical problems to solve set of especially in problems for the robustness of the approach in [6].

The design of the artificial bee colony (ABC) algorithm is modified a hybrid variant of a swarm-

based metaheuristics. This is referred to Collective Resource-based ABC with Decentralized tasking (CRbABC_Dt) in [29]. The proposed algorithm combines the attraction of de-centralization from neighbourhood information for wider exploration of search space.

A population-based search algorithm takes its name from the behaviour of honey bee colonies discovering new areas for the food foraging in natural land. The proposed algorithm is inspired by Bees algorithm**.** The behaviours of honey bee colonies look for the best solution for optimization problem. Each candidate solution is about a flower (food source) to define search the solution space, a colony (population) and bees (*n* agents). This algorithm evaluates fitness (profitability), and lands on a flower (solution) at each time. The algorithm performs a kind of neighbourhood for search combined with global search and has been successfully applied in various optimization areas for both combinatorial optimization and continuous optimization in order to solve many difficult optimization problems. The proposed algorithm is demonstrated the effectiveness and specific abilities for optimization solution in [7].

As it was mentioned above different multi-objective evolutionary algorithms such as bees algorithm, ant colony algorithm, genetic algorithms (GAs), particle swarm optimization (PSO) and differential evaluation algorithm have been designed. These algorithms have found many practical applications. Different evolutionary algorithms have been applied optimization issues successfully in order to solve many difficult optimization search problems. Many improvements have been done in order to develop optimization for search the best solution of the problems.

There have been many studies on finding new methods for optimization. Recently the field of combinatorial optimization has witnessed with the metaheuristic methods. Most of them are based on natural scientific happening. The development of new method is metaphor or man-made process increasing use of metaphors as inspiration and justification to lead the area of metaheuristics away from scientific. For more details are referred in [16]. In this study paper presents specify that the capacity MAs for unravelling various sorts of computerized IIR channel outlines is still restricted in [22]. Within a framework of two-step set memorization algorithm (TSMA) is designed to synthesize the advantages of evolutionary global research and local search techniques. The first one is about a competition is held among the candidate local search procedures. Its significant thought is to pick the best nearby inquiry system and to acquire great introductory state. The second one is about implement effective adaptive MA pursues the best solution.

Many research methods desire to find the best solution for global problems and increase the accuracy of the optimization. In this study paper, a novel Hypercube Optimization Search (HOS) algorithm which references the best elements from an extreme function to solve applied multivariate systems for optimization is designed in [3, 13]. This algorithm approaches in optimization which gets the best optimization elements to minimize (or maximize) since *min f(x) =-max (-f(x))*.

In this study paper, the performance of the HOS algorithm will be tested on test suite functions and some benchmark functions. The general performance of the algorithm will be evaluated in terms of solution accuracy and then the proposed algorithm will also be applied a timetabling problem. Experimental results have demonstrated the efficiency of the proposed method will be presented in terms of optimization accuracy.

2. The Design of Hypercube Optimization Search Algorithm

The structure of the HOS algorithm is used basic processes are described. The flowcharts of HOS algorithm mainly presents *Fly process* (initialization-evaluation), *Displacement-Shrink process* and *Searching Areas process* are described in Figure 1. The functions of each block are described and mathematical formulas and basic operations used in each processes are presented. The HOS algorithm is inspired by the behaviour of pigeon to discover new areas for seeds (food) in natural life. The behaviour of pigeon (flying) searches a moving for new locations of seeds. The behaviour of pigeon moves its around and contracts according to specific rules. In fact, the aim of pigeon discovers new places (areas) for seeds like as in natural life in [31]. In such behaviour, a moving pigeon searches for new locations of seed. The pigeon moves 360^0 swivel its around for in a unique way for mark areas and seeds. The possibility of a higher accumulation of location of seeds (or food) may be found in undiscovered

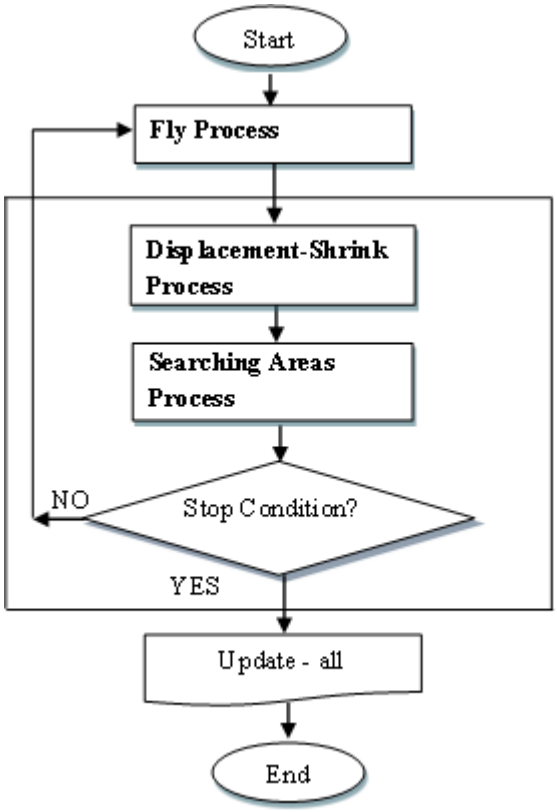

Figure 1. *The structure of Hypercube Optimization Search Algorithm*

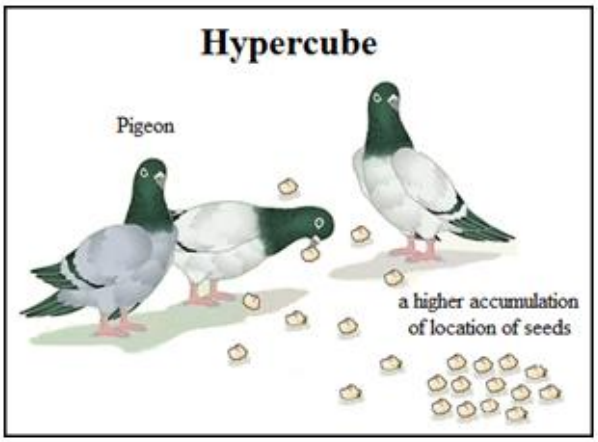

Figure 2. *The behaviour of pigeon is simulated to searching seeds in real life*

branches of the marked area. We have distributed the seeds on some area with different distribution a higher accumulation. In one of the point of this area, we had sown seeds with a higher accumulation of location of seeds (maximum). We detect that when a pigeon looking for the seeds in the area it does not directly moving (fly) to the area having a higher accumulation of location of seeds. The pigeon is starting from some arbitrary point time by time shrinkage the area and at the end come to the higher accumulation of location that has maximum seeds.

In a search process, the pigeon is not limited to a single area. The pigeon picks new search area according to the higher accumulation of location of seeds. The pigeon stops moving and keeps in mind the area which has seeds. After eating the seeds, the pigeon looks for a new search area. The pigeon moves another area branch to find a new area. The pigeon does not move to another area when it gets to an area that has the most seeds. For this investigation, some experiments have been done with the pigeons in real life. The behaviour of pigeon is simulated to searching seeds in Figure 2 in real life. In this study, the hypercube is used to describe the "searching area". Inside the search area, the objective function is evaluated at each solution. Beginning at the root area, any solutions which help us are 'neighbour' solutions. The value of an objective function is evaluated according to the quantity and accumulation of location of seeds. Then, the functional distances between each of the two solutions are determined evaluating best solution (lowest distances) for the next new search area. The HOS algorithm is mainly consist of processes that are given following below.

2.1 Fly Process (Initialization-Evaluation)

As mentioned above the HOS algorithm simulates the behaviour of a pigeon discovers new marking of areas. These marking of areas are performed and the pigeon is looking for new one after eating the food. By means of evaluating in the objective function in a given hypercube, the pigeons' positions are changed step by step from the initial points. Initial points in the process of fly are given in Table 1 in detail. The HOS algorithm begins the initialization of matrices and variables for the generation of a hypercube. The hypercube is represented by its centre (x_c) and radii (side R). Then the X matrix is

Table 1. *Initial points in the process of fly.*

Symbol	Definition
m	Dimension of hypercube
R	Radii of hypercube
$\boldsymbol{x} = \boldsymbol{x_i}$	take initial each Hypercube's position
LB, UB	lower and upper bounds
N	number of points
X	N x m points, solutions
F	N x 1 points, Values of functions
Create matrices:	X (N x m) F (N x 1) F_{BEST}: BEST VALUE

generated within the boundaries of the hypercube. The search area is bounded with the sides of a hypercube. The size of X solution is defined by (Nx1).

Initialization and evaluation is the first block of the HOS algorithm. The starting conditions are:

1. Lower (LB) and upper (UB) boundaries are used to generate the hypercube for all points. Initial points (x_i) are generated inside hypercube.

$$x_i = (x_{i1}, x_{i2}, \ldots, x_{im})$$

2. Points (x_i) are generated inside hypercube for it is generated best point (x_{best}) randomly is evaluated initialization of solutions according to the objective function.

$$F_i = (F_{i1}, F_{i2}, \ldots, F_{im})$$

The proposed algorithm is based on the generation and a uniform distribution of N searching points inside the initial hypercube. N should be high, that is the data points should be sufficiently dense so as to probe all the possible zones. Otherwise, the algorithm can take as the best zone (global optimum) a simply better one (local optimum). It is generated using point of values according to the objective function are determined. The idea is to have an approximate knowledge of where the lowest values of F. As pointed out above, a higher accumulation (and hence the number of points N) is a function of the dimension *m* of the hypercube. Problems with higher dimensions will require higher N, while smoother functions (or low dimensional) can be sampled with lower N.

As mentioned the hypercube is represented by the centre and radii. At the beginning, the value of radii of first HC is determined according to the change interval of objective (test) function. The initial point x_0 is generated as the centre of the first Hypercube (HC). Boundaries of HC determine the boundaries of search space. Using the value of HC's centre x_0 the dimensions of the hypercube are derived according to formula (2.1.1). In the next iterations, the proposed algorithm updates centre and radii of the hypercube in the displacement-shrink process. As a result of these operations the size of hypercube is reduced and search space is shrunk correspondingly. Note that in next iterations (i=2,3…) the parameters of the hypercube are updated using the values of X matrix. We have illustrated the fly process as follows with data points to create them with default values;

1. Dimension of hypercube

$$\boldsymbol{m} = \text{length}(x_0) \qquad (2.1.1)$$

2. Row vectors with lower and upper boundaries of HC

$$\text{LB} = min(X \text{ bounds}),$$
$$\text{UB} = max(X \text{ bounds}) \qquad (2.1.2)$$

3. Distance between ***m***-dimensional HC's

$$\text{D} = \text{UB} - \text{LB} \qquad (2.1.3)$$

4. Central Values

$$x_c = (\text{LB} + \text{UB}) / 2 \qquad (2.1.4)$$

5. Vector with radii of HC

$$R_0 = \text{D}/2; \quad \text{R} = R_0 \qquad (2.1..5)$$

In the following iterations, according to the matrix X, the row vector with the lower and upper boundaries (2.1.2) is determined. Using these boundaries, obtained from the first hypercube, the center points (2.1.4) and radii (2.1.5) of the next hypercube are determined. HOS algorithm begins with the initialization of matrices and variables, it goes to the main loop, by which convergence towards the global minimum is sought. The details regarding the visualization of flow-chart for Fly process of the HOS algorithm is illustrated in Figure 3.

In next iterations (i = 2, 3…) each position of the pigeon is created using the values of *X* matrix. These points form the new X_{new} matrix. This matrix is used to evaluate the test functions. Following evaluation, the best (minimum) value of function F_{best} and the corresponding X_{best} points are determined.

The X_{best} point is improved (updated) using local search, that is $X_{best}^{new} = X_{best} + \rho\Delta F$. Here $0 \le \rho \le 1$, F is the objective function. The improvement is continued until ΔF becomes acceptable small value-less than present value (tolF). The derived best points are used to determine the centre and radii of next position points. This operation is realised by calculating the mean of the centre of last position point ($X_{\text{last\_centre}}$) and last best (X_{best}) point. This process is called "displacement", which is described in next section.

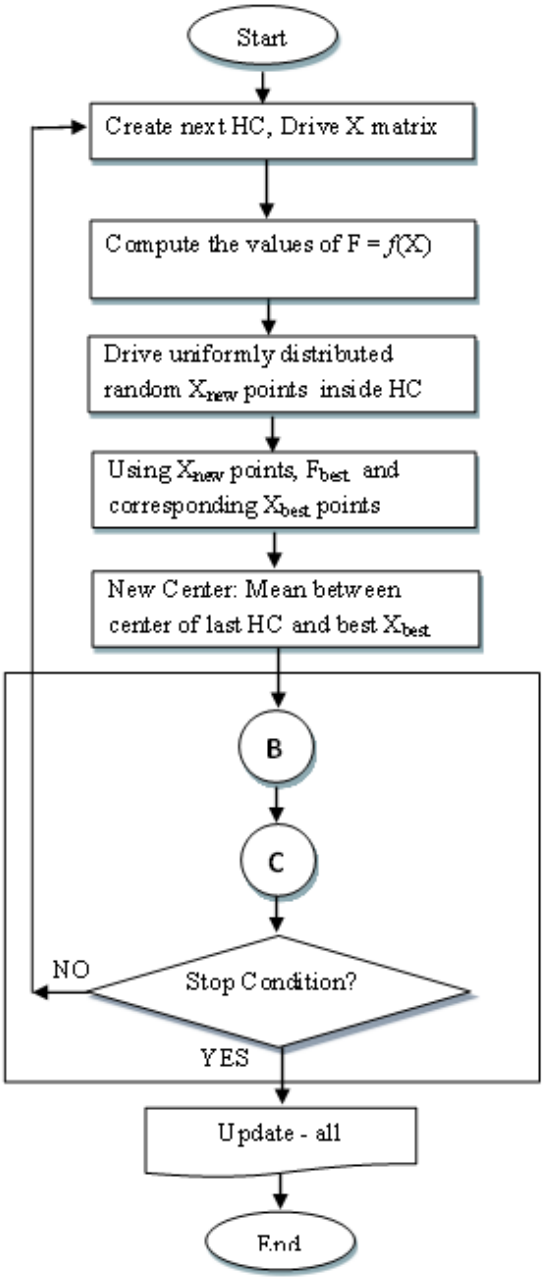

Figure 3. *The flow-chart for process of Fly*

2.2 Displacement-Shrink Process

The center of the next hypercube is calculated as the average between the best current point and the center of the previous hypercube. The average is about between the two values; they are considered to be a conservative criterion for avoiding excessive fluctuations in the search and avoiding drastically moving to a neighbouring area where a lower value has been found. The center and the radii of the new hypercube are determined as:

$$X_{new.centre} = \frac{X_{last\_centre} + X_{best}}{2}$$

$$R_{new} = R_{old} * S \qquad (2.2.6)$$

S is a factor of convergence which is defined in the next section (*see* equation (2.3.11)). As shown the calculated second position point is derived from the previous position point and the sizes of the second position point will be less than the sizes of the previous one. The details regarding the visualization of the flow-chart *Displacement-Shrink process* of the HOS algorithm is illustrated in Figure 4.The points in the HOS algorithm do not move according to some rule, nor does the method have memory of them, except for the best points X_{best}. New data points are generated and evaluated according to the objective functions. The hypercube size is changed according a result of this process the size of hypercube is reduced and the search space is shrunk correspondingly. The decrease in size of the hypercube allows an increase in a higher accumulation of the test points. This leads to the quick finding of the optimum value of the function. The idea of this is that if the best value of X moves significantly, then the global minimum is probably not found yet, and we are far from last phase of convergence. The number of generated points makes the deep search more efficient. This is shown to organize by increasing the number of test points in a search space by improving the best points using local search also. The algorithm evaluates the test functions in many points such an approach allows a rapid selection of a new best zone for an intense search in it. In addition to displacement, the HOS must shrink, in order to refine the search and converge towards a unique and certain minimum - global. This contraction is governed by the movement of the average of the best values. For large displacements, there is no contraction, because we interpret that the total minimum is still very uncertain. For small displacements or null displacements, the points will shrink as we approach the total minimum: the contraction is more important for the small movements. This guarantees rapid convergence of the method, while protecting against

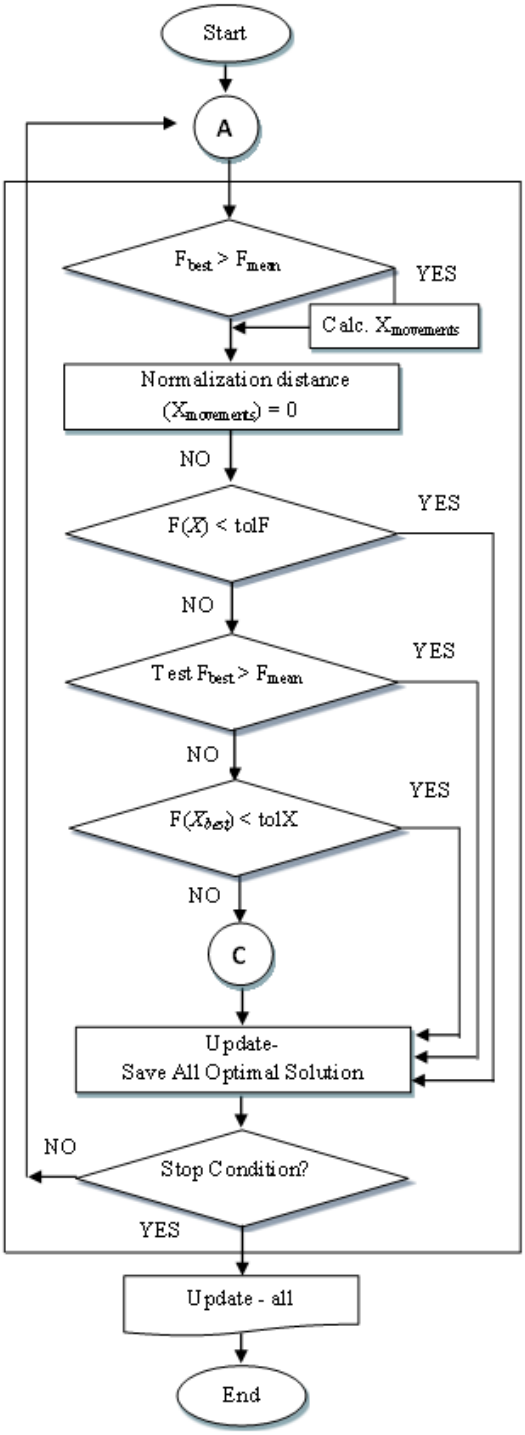

Figure 4. *The flow-chart for process of Displacement-Shrink*

blocking to an undesirable (local) minimum. The HOS will pass a set of points from the current position which determines the maximum distance. The ranges of displacements are shown below;

Step 1: Normalized ***x*** (the previous ***x*** for minimum)

$$x_n = \frac{(x - x_c)}{D} \qquad (2.2.7)$$

Step 2: Normalized $x_{\min}$ (current ***x*** for minimum)

$$x_{\min n} = \frac{(x_{\min} - x_c)}{D} \qquad (2.2.8)$$

Step 3: Normalized distance

$$d_n = \frac{sum((x_n - x_{\min n})^2)^{0.5}}{D} \qquad (2.2.9)$$

Step 4: Re-normalized distance

$$d_{nn} = \frac{d_n}{\sqrt{m}} \qquad (2.2.10)$$

For each iteration, the ***x*** displacement is computed and normalized twice: first each element of ***x*** is divided by the corresponding initial range (and thus the displacement is transformed into a unity-sided points) (***Step 1*** and ***Step 2***), and then that quantity is normalized again, dividing it by the diagonal of points such as $\sqrt{m}$ (***Step 3*** and ***Step 4***).

2.3 Searching Area Process

The search process is determined by computing the distances between old optimum (minimum) and new optimum values determined by equations (2.2.7 – 2.2.10) in particular equation (2.2.10). The *Searching area process* controls the movements of X according to the interval defined (particularly for $d_{nn} < 0.1$). The details regarding the visualization of flow-chart *Searching area process* of the HOS algorithm is illustrated in Figure 5. If the movement of X satisfies the condition then a factor of convergence S is calculated and updated it for each iterations:

$$S = 1 - 0.2e^{-3d_{nn}} \qquad (2.3.11)$$

where d_{nn} is computed by (2.2.10) and describes the normalized distance moved by the average of the last two best values of *X*. Next the update of solutions will be performed. The size (in all the dimensions) of the hypercube is reduced by multiplying with this factor, as mentioned in the previous section. The whole process is repeated until specific termination conditions are satisfied. When the new hypercube is set (centre and size), the function is again evaluated

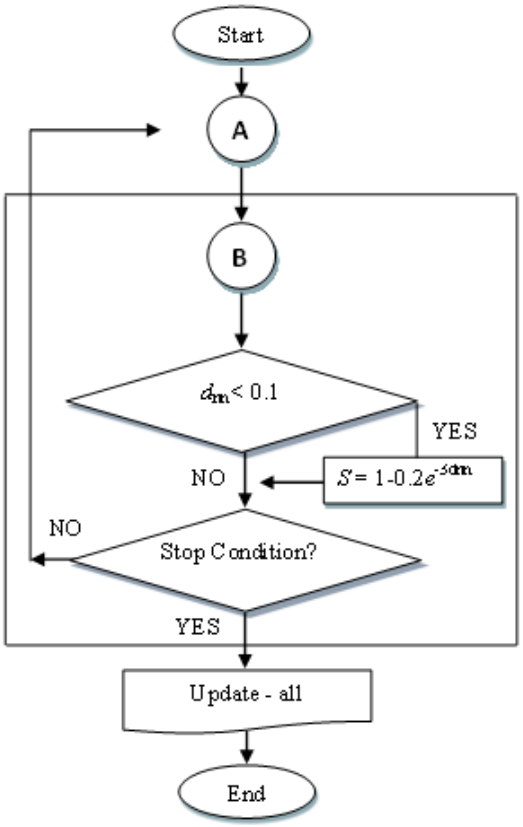

Figure 4. *The flow-chart for process of Searching area*

at new points, randomly (with uniform distribution) chosen inside the hypercube. Again, the new minimum is found and compared to the last minimum. If new minimum is worst (greater) than the previous one, d_{nn} is considered null and a new iteration will be done. If the same is found for several consecutive times the algorithm ends, and the best minimum is considered the global minimum. If the solutions found are the nearly the same then for obtaining exact solutions "the determination new position point and searching solution" process will be repeated 50 times. If in all iterations obtained values of d_{nn} that are the same, the process will be stopped. In other case the search of optimal solutions will be repeated. Scaling (the search space dimensionality) is implemented by reducing or dividing (distances) the initial size of HC (delimited by bounds) at each iteration. The movement of the focus (centre of next HC) and the new sizes of HC are computed from that scale. As shown in the above sections the HOS algorithm finds best points in given hypercube (search space) and then using this point draws a new hypercube, and again finds the next best point and creates the next new hypercube and so on. In this way the algorithm continues step by step approaches to the optimum point. This approach can be expressed as a hypercube evolution driven by convergence.

The HOS algorithm can be described as above. Input parameters for the algorithm are as follows (Figure 3.2): Objective function (defined as test function), initial lower and upper bounds of hypercube (absolute bounds) determined using test function, initial values of points (initial solution X_0), tolerances for function f and solution X (tolF, tolX). Output of the algorithm is the best individuals X_{best} in the final hypercube (HC).

3. Test Functions

The test functions used in simulation of the HOS algorithm are described. We have used five benchmark and eleven test suite functions are given in the literatures [26-30,33-49] to test the performance of the HOS algorithm and make a comparative analysis with other evolutionary algorithms. The benchmark function is general known described. These functions are summarized as follows; Rosenbrock function (f_1) is unimodal for lowest dimension as 2D and 3D but may have multiple minima in high dimensional problems. Schwefel 2.26 (f_2), Rastrigin function (f_3), Ackley function (f_4) and Griewank function (f_5) are multimodal. Next one is about test suite functions (Special Session and Competition on Real-Parameter Optimization). These test suite functions including; four unimodal functions and composition functions. It consists of eleven test suite functions that can be grouped into two types.

- Unimodal Functions (1 to 3)
- Composition Functions (4 to 11)

All test suite functions are minimization problems as $\text{Min } f(x)$, $x = [x_1, x_2, ..x_D]^{\text{T}}$. Here D is the dimension, $o_{i1} = [o_{i1}, o_{i2}, ..o_{iD}]^{\text{T}}$ are the shifted global optimum which is randomly distributed in $[-80,80]^D$. These functions are scalable and shifted to *0*. For convenience, the same search ranges are defined for all test functions from test suite functions for search range is $[-100, 100]^D$. Rotation matrix $\mathbf{M_i}$ is assigned to each test suite functions and with condition number c (Gram-Schmidt ortho-normalization) that is equal to 1 or 2. For a more detailed description of these test functions, the reader is referred [36] in details.

3.1 Simulation on Test Functions

In this section, the simulation of the Hypercube Optimization Search (HOS) algorithm has been performed. The performance of the proposed algorithm is tested using five benchmark functions and test suite functions are given in the previous section.

Table 2. *Simulation Results for Some Benchmark Functions for Dimension of 30.*

Functions (30*D*)		**Results**	**FEs**
Rosenbrock (f_1)	Best	0	1500
	Mean	0	
	StD.	0	
Schwefel 2.26 (f_2)	Best	3.82e-04	10^3
	Mean	3.82e-04	
	StD.	0	
Rastrigin (f_3)	Best	0	750
	Mean	0	
	StD.	0	
Ackley (f_4)	Best	8.88e-16	1500
	Mean	8.88e-16	
	StD.	8.96e-31	
Griewank (f_5)	Best	0	750
	Mean	0	
	StD.	0	

The proposed algorithm is applied to minimize a set of problems of dimensions as 30*D*, 50*D* and 100*D* for five benchmark functions. The proposed algorithm has been set the population size equal to 100 for the cases of 30*D* and 50*D*, and 400 for the case of 100*D*, respectively. It is applied to minimize a set of problems of dimensions as 10*D* and 30*D* for test suite functions. At first, the performance of the HOS algorithm is statistically applied to minimize a set of five benchmark functions of dimensions as

Table 3. *Simulation Results for Some Benchmark Functions for Dimension of 50.*

Functions (50*D*)		**Results**	**FEs**
Rosenbrock (f_1)	Best	0	1500
	Mean	0	
	StD.	0	
Schwefel 2.26 (f_2)	Best	6.36e-04	10^3
	Mean	6.36e-04	
	StD.	0.00e+00	
Rastrigin (f_3)	Best	0	750
	Mean	0	
	StD.	0	
Ackley (f_4)	Best	8.88e-16	1500
	Mean	8.88e-16	
	StD.	8.96e-31	
Griewank (f_5)	Best	0	10^3
	Mean	0	
	StD.	0	

Table 4. *Simulation Results for Some Benchmark Functions for Dimension of 100.*

Functions (100D)		**Results**	**FEs**
Rosenbrock (f_1)	Best	0	2. 10^3
	Mean	0	
	StD.	0	
Schwefel 2.26 (f_2)	Best	1.27e-03	10^3
	Mean	1.27e-03	
	StD.	0.00e+00	
Rastrigin (f_3)	Best	0	10^3
	Mean	0	
	StD.	0	
Ackley (f_4)	Best	8.88e-16	1750
	Mean	8.88e-16	
	StD.	8.96e-31	
Griewank (f_5)	Best	0	10^3
	Mean	0	
	StD.	0	

30*D*, 50*D* and 100*D*.
We have summarized the mean and standard deviation of the results obtained by the HOS algorithm that tested functions evaluations over successful 50 runs.

The details regarding the visualization of benchmark function results and the number of function evaluations (FEs) are shown in Table 2 for dimenson of 30, Table 3 for dimenson of 50 and Table 4 for dimenson of 100. Next section is about simulation of these benchmark function that are given figures as dimension of 30, dimension of 50 and dimension of 100.

3.1.a Simulation for Rosenbrock Function

The HOS algorithm is applied for optimization of test function (f_1) having 30*D*, 50*D* and 100*D* dimensions that are given following figures. f_1 was obtained as 0 (a), 0 (b) and 0 (c) after 1500, 1500 and 2000 evaluations respectively.

3.1.b Simulation for Schewel Function 2.26

The HOS algorithm is applied for optimization of test function (f_2) having 30*D*, 50*D* and 100*D* dimensions that are given the following figures. The minimum of Schwefel function 2.26 was obtained as 3.82e-04 (a), 6.36e-04 (b) and 1.27e-03 (c) after 1000 evaluations for these dimensions respectively.

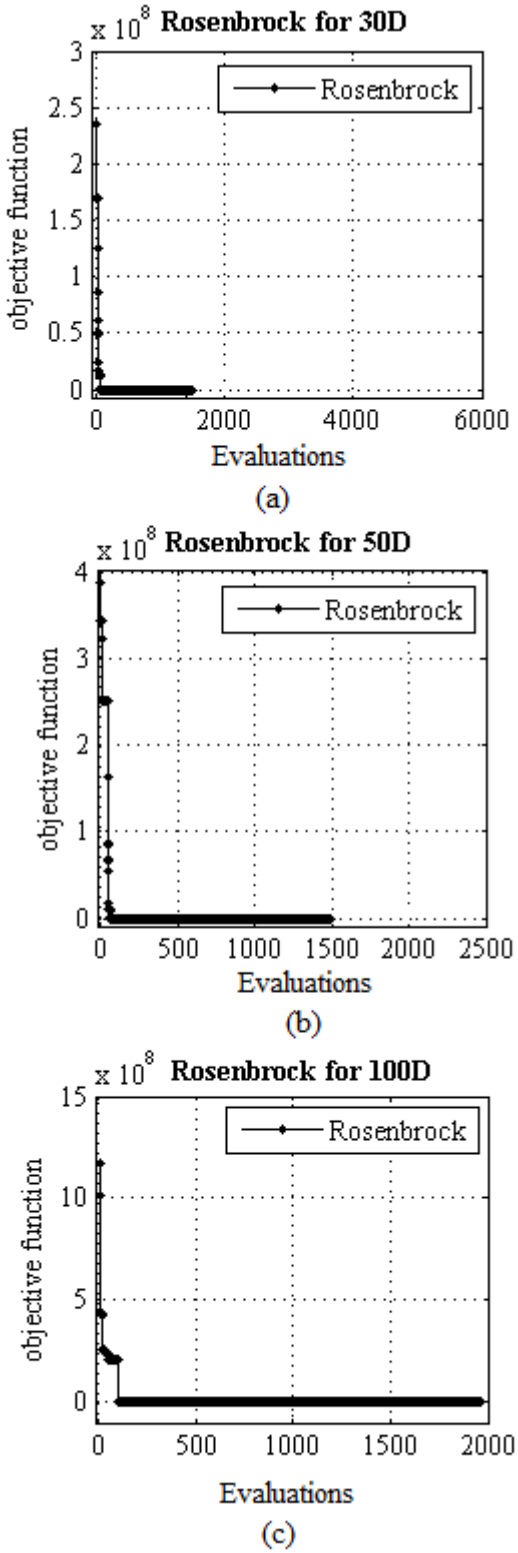

Figure 5. *Rosenbrock for Dimension of 30, 50 and 100 for (a), (b) and (c) respectively*

3.1.c Simulation for Rastrigin Function

The HOS algorithm is applied for optimization of test function (f_3) having 30*D*, 50*D* and 100*D* dimensions that are given in figures. Rastrigin function was obtained as 0.00e+00 (a), 0.00e+00 (b) and 0.00e+00 (c) after 750, 750 and 1000 evaluations respectively.

3.1.d Simulation for Ackley Function

The HOS algorithm is applied for optimization of test function (f_4) having 30*D*, 50*D* and 100*D* dimensions that are given in figures. The minimum of *Ackley function* was obtained as 8.88e-16 (a), 4.44e-15 (b) and 8.88e-16 (c) after 1500, 1500 and 1750 evaluations respectively.

3.1.e Simulation for Schewel Function 2.26

The HOS algorithm is applied for optimization of test function (f_5) having 30*D*, 50*D* and 100*D* dimensions that are given in figures. The minimum of *Griewank function* was obtained as 0.00e+00 (a), 0.00e+00 (b) and 0.00e+00 (c) after 750, 1000 and 1000 evaluations respectively.

3.2 Simulation on Test Suit Functions

The proposed algorithm is applied to minimize a set of eleven problems of dimensions 10*D* and 30*D* after setting the maximum function evaluations (FEs) equal to $D*10^4$ for all test suite functions.

Table 5. *Simulation Results for Some Test Suit Functions for Dimension of 10.*

Functions (10D)	**Mean**
(f_6) Rotated High Conditioned Elliptic	0
(f_7) Rotated Bent Cigar	0
(f_8) Rotated Discus	2.44e-06
(f_9) Composition 1 (N=5)	1.35e+02
(f_{10}) Composition 2 (N=3)	6.79e+00
(f_{11}) Composition 3 (N=3)	3.47e+01
(f_{12}) Composition 4 (N=5)	9.86e+00
(f_{13}) Composition 5 (N=5)	6.67e+01
(f_{14}) Composition 6 (N=5)	1.29e+02
(f_{15}) Composition 7 (N=3)	3.49e+02
(f_{16}) Composition 8 (N=3)	6.69e+01

We have summarized the mean and the standard deviation of the results obtained by the HOS algorithm in results of evaluations of test functions over successful 50 independent runs. The details regarding the visualization of the experimental results of test suite functions are shown in Table 5 for dimension of 10 and in Table 6 for dimension of 30. In Table 5 and in Table 6, these experimental results

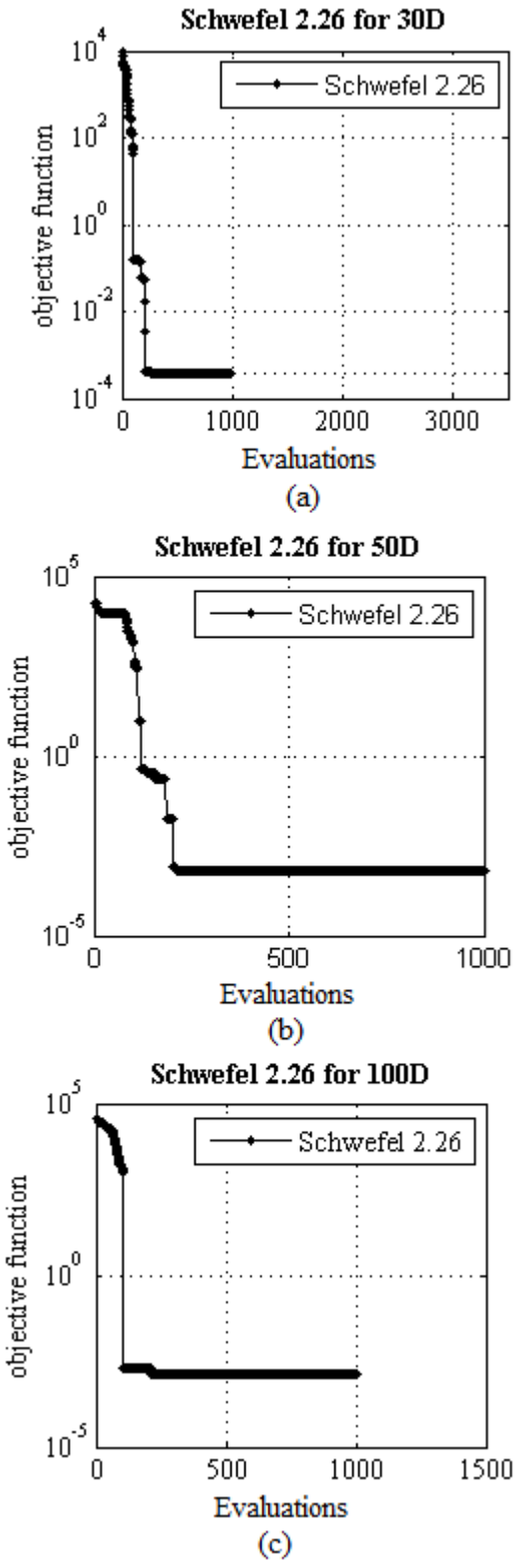

Figure 6. *Schewel 2.26 for Dimension of 30, 50 and 100 for (a), (b) and (c) respectively*

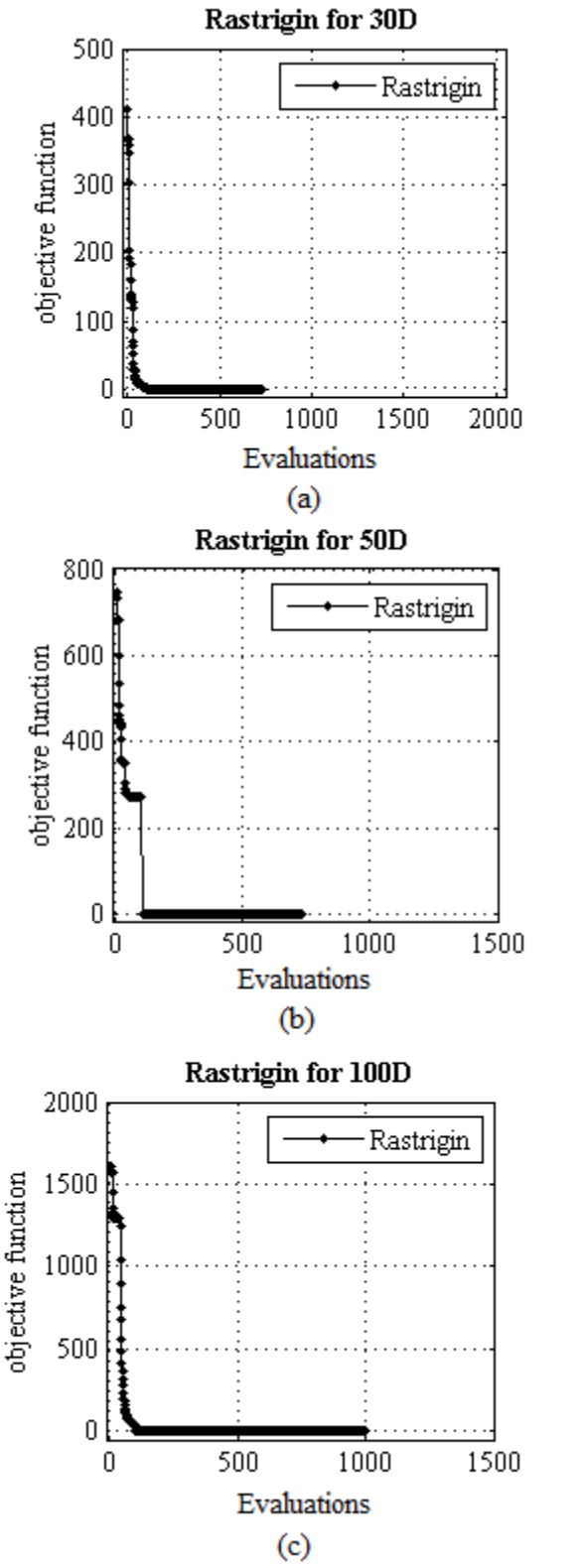

Figure 7. *Rastrigin for Dimension of 30, 50 and 100 for (a), (b) and (c) respectively*

suggest that the overall convergence rates of the HOS algorithm still perform better for the set of problems. We can see that by increasing learning iterations, the accuracy of the HOS algorithm is increased for these functions.

3.3 Comparative Results

This section presents a comparison of the performances of the HOS algorithm with different evolutionary algorithms by using some Benchmark

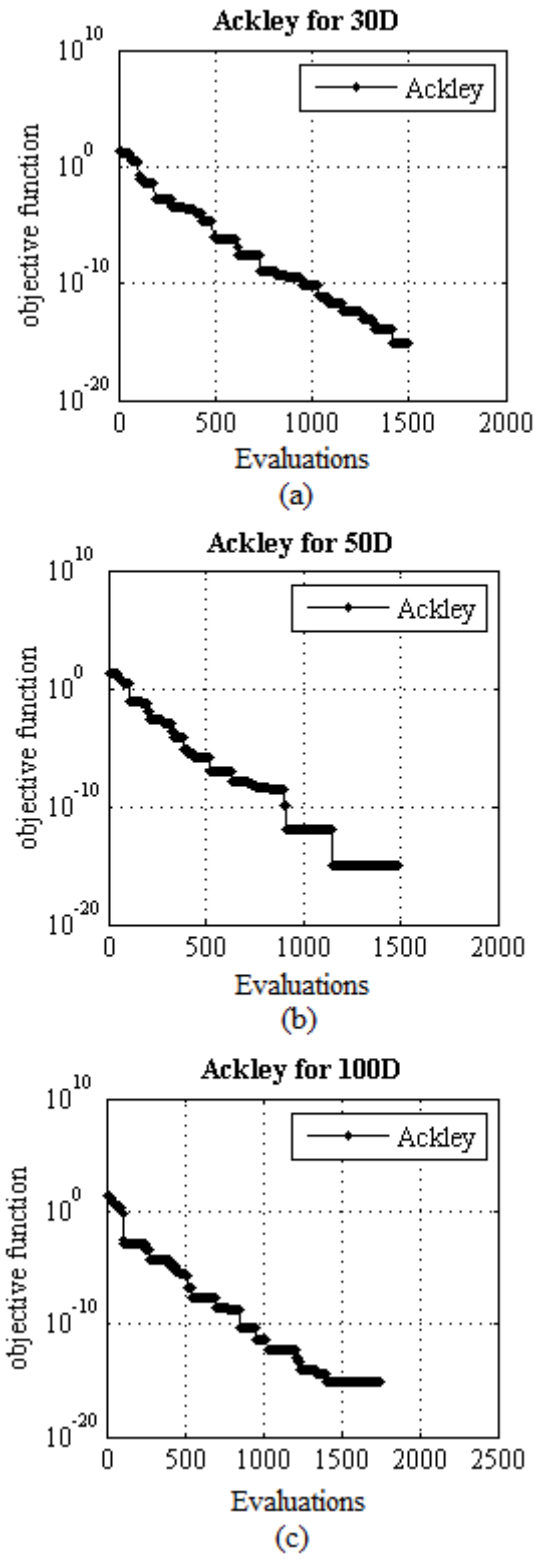

Figure 8. *Ackley for Dimension of 30, 50 and 100 for (a), (b) and (c) respectively*

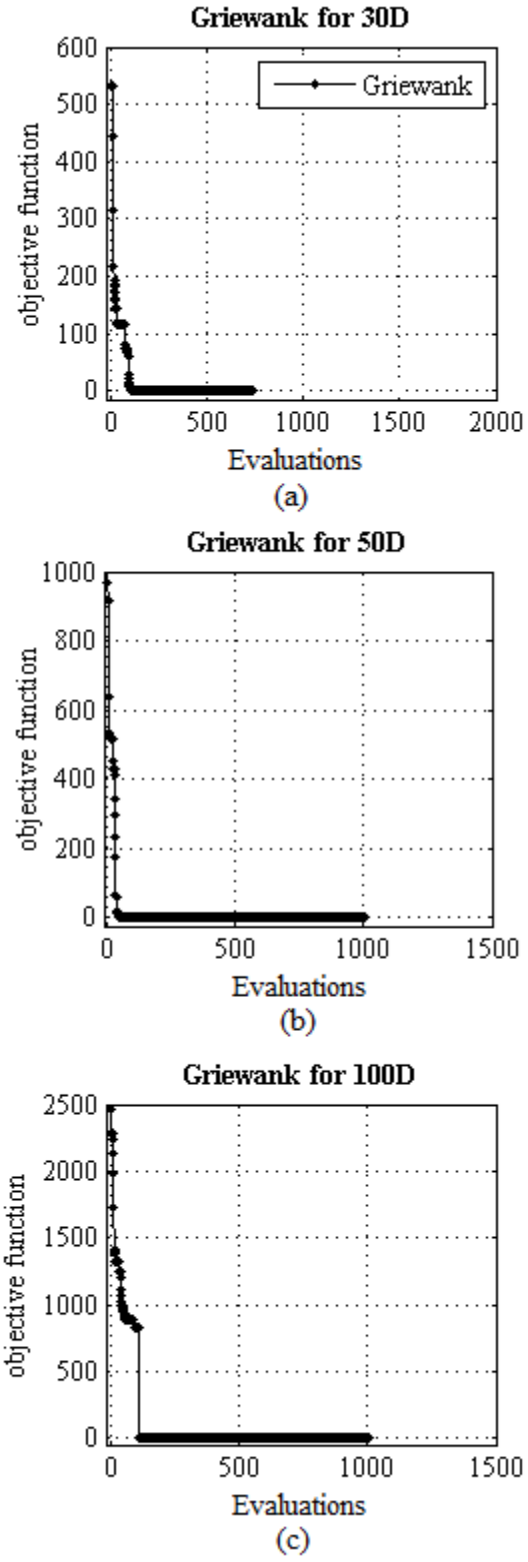

Figure 9. *Griewank for Dimension of 30, 50 and 100 for (a), (b) and (c) respectively*

Functions and test suite functions given in above. In this study, statistical tests are provided in order to show statistical differences between simulation results of the HOS algorithm and the simulation results of other evolutionary algorithms.

All comparisons of the test optimization search methods defined same initial conditions, parameters as population size; 100 in the case of 30*D* and 400 in the case of 100*D* [28]. At the first one, dimension is set to 30*D*, the population size is set at 100 for these

Table 6. *Simulation Results for Some Test Suit Functions for Dimension of 30.*

Functions (30D)	**Mean**
(f_6) Rotated High Conditioned Elliptic	0
(f_7) Rotated Bent Cigar	1.89e-06
(f_8) Rotated Discus	0
(f_9) Composition 1 (N=5)	2.65e+02
(f_{10}) Composition 2 (N=3)	8.89e+01
(f_{11}) Composition 3 (N=3)	1.55e+02
(f_{12}) Composition 4 (N=5)	4.46e+01
(f_{13}) Composition 5 (N=5)	1.13e+02
(f_{14}) Composition 6 (N=5)	2.36e+02
(f_{15}) Composition 7 (N=3)	2.65e+02
(f_{16}) Composition 8 (N=3)	4.15e+02

function evaluations over successful 50 runs. The mean and the standard deviation of the results obtained by each algorithm for five Benchmark functions that are summarized in Table 2, in Table 3 and in Table 4. The performances for all algorithm results are statistically shown the better one that is highlighted in bold. For each Benchmark function, the results of above mentioned two, namely; DE and jDE algorithms are compared and the better result (whose mean's value is less) is selected for statistical comparison with the results of the HOS algorithm. This idea is used in all future comparisons. The performances of HOS algorithm demonstrate much better than others to be marked in bold given in Table 7 in detail.

Table 7. *Simulation Results for Some Benchmark Functions for Dimension of 30.*

Functions		**HOS**	**DE**	**jDE**	**FEs**
f_1	Mean	**0**	2.1e+00	1.3e+01	3.10^3
	StD.	**0**	(1.5e+00)	1.4e+01)	
f_2	Mean	3.8e–04	5.9e+03	**7.9e–11**	10^3
	StD.	(0.0e+00)	(1.1e+03)	**(1.3e–10)**	
f_3	Mean	**0**	1.8e+02	1.5e–04	10^3
	StD.	**0**	(1.3e+01)	(2.0e–04)	
f_4	Mean	**8.8e–16**	9.7e–11	4.7e–15	2.10^3
	StD.	**(8.9e–31)**	(5.0e–11)	(9.6e–16)	
f_5	Mean	0	0	0	3.10^3
	StD.	0	0	0	

The HOS algorithm works much better than others for f_1, f_3, f_4 and f_5. The performances of DE and jDE are better for f_5 also. The performance of jDE performs better only for f_2. At the second one, the performance of the HOS algorithm is also compared with the performances of algorithms namely; PSO algorithms, UPSO (unified particle swarm optimizer), CPSO-*H* (a cooperative particle swarm optimizer) and CLPSO (comprehensive learning particle swarm optimizer) [27].

These results suggest that the overall performance of the HOS algorithm is better for the set of benchmark functions for f_1, f_3 and f_5; (be marked in boldface) that are equal to 0 (zero). The HOS algorithm is applied for the minimization of f_2 and f_4 that are obtained as 3.82e-04 and 8.88e-16 correspondingly. The performance of CLPSO-*H* is performed better on f_3 *also* and CLPSO is performed better than others for f_2 and f_4.

In Table 8 is shown the comparative results of the algorithms; the performance of the HOS algorithm is generally much better than the performances of UPSO, CPSO-*H* and CLPSO algorithms.

In the third stage, the performance of the HOS algorithm has been compared with algorithms, namely; *CRb*ABC, m-ABC and CLIPSO from eleven test suite functions [29]. The HOS algorithm is also evaluated by considering the cases in which the problem dimension is a set of dimensions. At first problem dimension is set as 30*D*, the maximum function evaluations (FEs) is set at $D*10^4$ for all test suite functions. The mean and the standard deviation of the results obtained by each algorithm for these functions are summarized in Table 9. These results suggest that the overall performance of the HOS algorithm is the best for the problems f_6-f_8, f_{10}-f_{13} and f_{15} (be marked in boldface). *CRb*ABC performed well on f_6, f_9, f_{14} and f_{16}. CLIPSO algorithms performed well on f_6 also. The comparative results have been demonstrated that the HOS algorithm has better performances on a set of problems. In the next stage, the dimensions are set as 100*D* for all specific five benchmark functions. The mean and the standard deviation of the results obtained by each

Table 8. *Simulation Results for Some Benchmark Functions for Dimension of 30.*

Functions		**HOS**	CLPSO	CPSO-*H*	UPSO
f_1	Mean	**0**	2.10e+01	7.08e+00	1.51e+01
	StD.	**0**	2.98e+00	8.01e+00	8.14e-01
f_2	Mean	3.82e-04	**1.27e-12**	1.08e+03	4.84e+03
	StD.	0.00e+00	**8.79e-13**	2.59e+02	4.76e+02
f_3	Mean	**0**	4.85e-10	0	6.59e+01
	StD.	**0**	3.63e-10	0	1.22e+01
f_4	Mean	8.88e-16	**0**	4.93e-14	1.22e-15
	StD.	8.96e-31	**0**	1.10e-14	3.16e-15
f_5	Mean	**0**	3.14e-10	3.63e-02	1.66e-03
	StD.	**0**	4.64e-10	3.60e-02	3.07e-03

Table 9. *Experimental Results for Test suite functions for Dimension of 30.*

f	**HOS**	**m-ABC**	**CLIPSO**	***CRb*ABC_Dt**

f_6	**0.00e+00**	8.57e+04	**0.00e+00**	**0.00e+00**
f_7	**1.89e-06**	2.12e+07	1.82e+08	3.52e+06
f_8	**0.00e+00**	1.87e+03	1.38e+04	1.17e+02
f_9	2.65e+02	4.43e+02	2.00e+02	**1.93e+02**
f_{10}	**8.89e+01**	3.16e+03	9.20e+02	3.03e+02
f_{11}	**1.55e+02**	4.61e+03	8.26e+03	4.34e+03
f_{12}	**4.46e+01**	3.15e+02	2.60e+02	2.23e+02
f_{13}	**1.13e+02**	3.04e+02	2.99e+02	2.86e+02
f_{14}	2.36e+02	4.06e+02	3.33e+02	**2.00e+02**
f_{15}	**2.65e+02**	1.41e+03	9.50e+02	5.70e+02
f_{16}	4.15e+02	**3.00e+02**	3.00e+02	**2.83e+02**

Table 10. *Experimental Results for Some Benchmark Functions for Dimension of 100.*

Functions		**HOS**	**DE**	**jDE**	**FEs**
f_1	Mean	**0**	9.5e+01	7.2e+01	6.10^3
	StD.	**(0)**	(1.4e+01)	(1.1e+01)	
f_2	Mean	**7.9e–02**	3.2e+04	4.9e+03	10^3
	StD.	**(1.2e–01)**	(4.7e+02)	(4.1e+02)	
f_3	Mean	**0**	8.6e+02	2.1e–04	3.10^3
	StD.	**(0)**	(2.2e+01)	(2.1e–04)	
f_4	Mean	**8.8e–16**	1.3e–01	9.9e–14	3.10^3
	StD.	**(8.9e–31)**	(2.4e–02)	(2.0e–14)	
f_5	Mean	**0**	2.0e–01	**0**	3.10^3
	StD.	**(0)**	(5.8e–02)	**(0)**	

algorithm are summarized in Table 10. The comparative results demonstrate experimental results for the performance of the HOS algorithm is much better than others for high dimensional optimization function (be marked in boldface).

Using test suite functions the simulation results of the HOS algorithm are compared with the results of the evolutionary algorithms given in research [37-49] which are referred in test suite functions. The comparative results as the mean and standard deviation for 10*D* and 30*D* are given in Table 11 and in Table 12 respectively. As shown in tables the results of the HOS algorithm has shown better performances than other algorithms in most of the functions. On 10*D* problem the HOS algorithm has performed better for f_9-f_{12}, f_{14} and f_{16}. The HOS has also shown good performance on f_6 and f_7. RSDE [47]– on f_6-f_8 and f_{15}. FCDE [44]-on f_7 - f_8 and f_{13}. GaAPADE [37] performed well on f_7. On 30*D* problems the HOS algorithm has performed much better for f_8-f_{16}. The HOS has also shown good performance on f_6. GaAPADE [39] - on f_6-f_8, FCDE [46] - on f_7, RSDE [49] – on f_7 also.

Table 11. *Comparative Results of 10D for Test Suite Functions*

***f* (*10D*)**		**HOS**	RSDE [47]	FCDE [44]	GaAPADE [37]
f_6	*Mean*	**0**	**0**	3.06e-04	1.3e+01
	StD.	**0**	**0**	2.18e-03	1.4e+01)
f_7	*Mean*	**0**	**0**	**0**	**7.9e–11**
	StD.	**0**	**0**	**0**	**(1.3e–10)**
f_8	*Mean*	2.44e-06	**0**	**0**	1.5e–04
	StD.	2.99e-06	**0**	**0**	(2.0e–04)
f_9	*Mean*	**1.35e+02**	3.29e+02	3.29e+02	3.29e+02
	StD.	**1.15e+01**	2.78e-13	2.66e-13	0.00e+00
f_{10}	*Mean*	**6.79e+00**	1.19e+02	1.38e+02	1.08e+02
	StD.	**2.82e-01**	6.59e+00	1.37e+01	2.08e+00
f_{11}	*Mean*	**3.47e+01**	1.30e+02	1.84e+02	1.68e+02
	StD.	**2.15e+00**	1.93e+01	2.07e+01	4.11e+01
f_{12}	*Mean*	**9.86e+00**	1.00e+02	1.00e+02	1.00e+02
	StD.	**2.27e+00**	3.65e-02	1.59e-01	1.74e-02
f_{13}	*Mean*	6.67e+01	9.12e+01	**2.55e+01**	9.56e+01
	StD.	5.75e+00	1.40e+02	**7.77e+01**	1.63e+02
f_{14}	*Mean*	**1.29e+02**	3.87e+02	4.90e+02	3.84e+02
	StD.	**3.42e+01**	4.88e+01	9.61e+01	3.36e+01
f_{15}	*Mean*	3.49e+02	**2.13e+02**	2.06e+05	2.22e+02
	StD.	1.16e-13	**2.59e+01**	7.25e+05	6.81e-01
f_{16}	*Mean*	**6.69e+01**	5.05e+02	8.87e+02	4.68e+02
	StD.	**2.59e+00**	1.06e+02	3.48e+02	1.90e+01

Table 12. *Comparative Results of 30D for Test Suite Functions*

***f* (*30D*)**		**HOS**	RSDE	FCDE	GaAPADE
f_6	*Mean*	**0**	1.50e+03	6.54e+04	**0**
	StD.	**0**	1.70e+03	4.90e+04	**0**
f_7	*Mean*	1.89e-06	**0.00e+00**	**0.00e+00**	**0**
	StD.	1.45e-06	**5.99e-09**	**0.00e+00**	**0**
f_8	*Mean*	**0**	4.74e-02	3.51e+01	**0**
	StD.	**0**	1.16e-01	1.26e+02	**0**
f_9	*Mean*	**2.65e+02**	3.15e+02	3.15e+02	3.15e+02
	StD.	**2.38e+01**	1.62e-06	1.67e-12	0.00e+00
f_{10}	*Mean*	**8.89e+01**	2.24e+02	2.50e+02	2.24e+02
	StD.	**1.08e+00**	1.65e+00	6.82e+00	8.64e-01
f_{11}	*Mean*	**1.55e+02**	2.03e+02	2.05e+02	2.03e+02
	StD.	**3.65e-01**	1.17e-01	2.29e+00	7.19e-02
f_{12}	*Mean*	**4.46e+01**	1.00e+02	1.01e+02	1.00e+02
	StD.	**1.17e+00**	4.14e-02	1.10e-01	2.40e-02
f_{13}	*Mean*	**1.13e+02**	4.69e+02	6.18e+02	3.19e+02
	StD.	**5.53e+00**	9.46e+01	2.32e+02	3.46e+01
f_{14}	*Mean*	**2.36e+02**	9.05e+02	1.50e+03	8.38e+02
	StD.	**9.42e+01**	1.21e+02	3.75e+02	2.96e+01
f_{15}	*Mean*	**2.65e+02**	6.52e+05	1.06e+06	7.17e+02
	StD.	**8.96e+01**	2.66e+06	3.28e+06	3.99e+00
f_{16}	*Mean*	**4.15e+02**	1.70e+03	2.53e+03	1.52e+03
	StD.	**2.40e+02**	8.67e+02	9.82e+02	8.02e+02

4. Application of HOS Algorithm for Timetabling Problems

In this chapter presents the real-life application of the HOS algorithm for timetabling problems. Exam timetabling problem have been considered and the problem is solved using the HOS algorithm. The timetabling problems seem to be difficult to solve finding optimal results for the scheduling of exams in institutions of higher education. The scheduling of examination timetabling problems consist of

allocating a number of exams to a limited number of periods with specific constraints related to the avoiding the overlapping of exams having students in common, satisfying room and time constraints, etc. These constraints may depend on certain limitations that are given the following general conditions:

- No students should have to take two exams in adjoining periods.
- No students should have to take two exams in one day.

For each period, these constraints mainly form the basis for a feasible scheduling of exams timetabling problem. For a more detailed description of timetabling, for the reader is referred in [50]. There are similar and differing characteristic problems for course scheduling and examination scheduling problems. In both problems for solving examination scheduling problems; students should have to take one examination or adjoining one period at a time. On the other hand, there may have a fixed time period in examination scheduling problems. This means that students should have to take one exam in one day so that students will be rested for the next exams. There are finding new methods and improvements for known evolutionary algorithms used in optimization. In this study, a novel Hypercube Optimization Search (HOS) method which references the best elements from an evaluation function to solve applied examination scheduling problems for optimization is designed. This method can be expressed as a hypercube evolution.

The performance of the HOS method tested on scheduling of examination timetabling problems, a set of evaluation functions implemented as MATLAB program. The proposed method used specified evaluation functions defined as following:

1. Every exam should be scheduled only once in the timetable.

$$\min \frac{\sum_{i=1}^{N-1} F(i)}{M} \qquad (4.1)$$

2. No clashing exams should be scheduled within the same period.

$$\sum_{i=1}^{E-1} \sum_{j=i}^{E} \sum_{p=1}^{P} T_{ip} T_{jp} C_{ij} = 0 \quad (4.2) \quad \text{where}$$

C_{ij} is defined both exams *i* and *j* (both exams).

3. All periods have limited seats. If a number of students taking exam for any period are less than the limited seats, it is available.

$$\sum_{p=1}^{P+1} T_{ip} S_i \le S \qquad (4.3) \text{ where } S_i \text{ is defined}$$

student taking exam *i*.

4. P (penalty) is based on function (evaluation) for a period. This is defined as below;

$$\text{Penalty} = \sum_{i=1}^{E-1} \sum_{j=i+1}^{E} T_{ip} T_{j(p+1)} C_{ij} D_{p(p+1)} \qquad (4.4)$$

Penalty is clearly relative for necessity something to compute the mean penalty per period of the best solution as defined following;

$$\text{Mean} = \frac{\text{The best of Fitness}}{\text{Total Periods}} \qquad (4.5)$$

For each period, the HOS method can compute a probability of being disrupted according to equation 4.6. This equation is shown how the probability of being disrupted is computed with bias being a definable value (0.1) to vary the probability of periods being disrupted. Periods reasoning greater than mean penalty (equation 4.5) are disrupted as below;

$$\text{Probabilit y} = \begin{cases} \text{if Penalty} \ge \text{Mean}, 1 \\ \text{if Penalty} < \text{Mean}, \dfrac{\text{Penalty} + (2 * \text{bias})}{2 * \text{Mean}} \end{cases}$$

(4.6)

The following experimental data is taken from *ftp://ftp.cs.nott.ac.uk/ttp/Data/* for modelling of timetabling problem. Experiments are carried out on an actual dataset from University of Carleton and University of King Fahd, University of Nottingham. In Table 13, the data used is represented by the following information:

carf92 University of Carleton (1992), Ottawa
kfu University of King Fahd, Dhahran
nott University of Nottingham, UK

The HOS method tested on a range of real data with the exception of varying to represent the real life

information. In this method is using exams (E) that must be scheduled in three periods (P) in one day. The performance of the Hypercube optimization search method is tested on three univerties data sets with sizes subset of 100. These data sets are tested with all the heuristics given 50 runs each and the average given results in Table 14.

Table 13. *Data set from the universities*

Data	**carf92**	**Kfu**	**Nott**
Period	32	20	23
Exam	543	461	800
Student	18 419	5 349	7 896
Enrolment	55 522	25 118	34 265
Students per Periods	2000	1955	1550
Conflict Density	0.14	0.06	0.03

A1: The Hypercube Optimization Search Method
A2: Multistage Evolutionary (ME) Method with Largest Degree
A3: Multistage Evolutionary (ME) Method with Colour Degree

Table 14. *Comparative results of applying dataset*

Functions	**2nd Order Same Day**	**2nd Order Overnight**	**Penalty**
carf92			
A1	298	725	1 836
A2	469	614	2 104
A3	359	773	2 034
Kfu			
A1	221	818	1 779
A2	222	838	1 811
A3	260	999	1 836
Nott			
A1	102	325	10 990
A2	155	416	14 984
A3	106	330	15 621

The focus of the experiments is demonstrates to the HOS method effects on quality when has been started on already better solutions than Multistage Evolutionary (ME) Method with Largest Degree and Multistage Evolutionary (ME) Method with Colour Degree for solving a set of optimal results in [51]. The finding of a near optimal solution of timetabling is important. Many timetabling process is a little more complicated when room was an allocation problem (maximum sitting students per periods)

5. Conclusion

The reviews of optimization search algorithms have shown that conventional optimization search algorithms sometimes cannot find a global optimum and has local optima problem. Evolutionary search algorithms can outperform conventional optimization algorithms and find a global optimum, without being trapped in local optima. Nowadays for many real world optimization problems the designing of evolutionary search algorithm with high optimization accuracy is more important. In this study, a novel evolutionary search algorithm to solve optimization of multivariate systems is proposed.

The basic processes of hypercube optimization search algorithm are designed. The HOS algorithm is a new intense stochastic search method based on a hypercube evolution. The algorithm comprises the *fly process*, *displacement-shrink process* and *searching areas process*. The design of the basic processes of HOS algorithm is presented and simulations of the algorithm have been carried out for global optimization of a set of CEC'14 test functions and benchmark functions. The performances of the HOS algorithm were also compared with the performances of other evolutionary algorithms. These are a new differential evolution (DE) algorithms as jDE; PSO algorithms including; unified particle swarm optimizer (UPSO), a cooperative particle swarm optimizer (CPSO-H) and comprehensive learning particle swarm optimizer (CLPSO); Artificial Bee Colony with *CRb*ABC_Dt and others. The comparative results have more generally been demonstrated that the HOS algorithm has shown better performance for optimization most of low and also high dimensional functions. The large number of generated points and population size allows it to quickly solve low and high dimensional optimization problems.

The proposed approach allows increasing the accuracy of the system. The simulation results of optimization problems have been demonstrated that the solution accuracy and success rate of the system has considerably increased and the algorithm has advantages over other well-known algorithms. The algorithm is also applied to the solution of a timetabling problem. The simulation of the HOS algorithm on test functions and timetabling problems have demonstrated the proposed optimization search algorithm that has shown to be a promising approach and is comparable with specialized algorithms to solve a set of global optimization real problems.

References

[1] Spall, J. (1998). An overview of the simultaneous perturbation method for efficient optimization. *Johns Hopkins APL Technical Digest, 19*, 482-492.

[2] Goldberg, D., E. (1989). Genetic Algorithms in Search, Optimization, and Machine Learning. Boston: Addison–Wesley Publishing Company.

[3] Mukhopadhyay, A., Maulik, A., Bandyopadhyay, S., & Coello, C., A., C. (2014). A Survey of Multiobjective Evolutionary Algorithms for Data Mining: Part II. *IEEE Trans. On Evolutionary Computation*, *18*(1), 20-35.

[4] Kennedy, J., & Eberhart R., C. (2001). Swarm Intelligence. San Francisco: Morgan Kaufmann PublishersInc.

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

[6] Dorigo, M., Maniezzo V., & Colorni A. (1996). The ant system: optimization by a colony of cooperative agents. *IEEE Transactions on System Man Cybernet, 26*, 29-41.

[7] Karaboga, D., & Basturk, B. (2008). On the performance of artificial bee colony (ABC) algorithm. *Applied Soft Computing*, *8*(1), 687-697.

[8] Beyer, H. & Schwefel, H. (2002). Evolution strategies-Acomprehensive introduction.*Natural Computing*, *1*, 3-52.

[9] Ni, H., & Wang, Y. (2013). Stock index tracking by pareto efficient genetic algorithm. *Applied Softcomputing*, *13*(12),4519-4535.

[10] Dhiman, R., & Priyanka, J., S. (2014). Genetic algorithms tuned expert model for detection of epileptic seizures from EEG signatures. *Applied Softcomputing, 19*, 8-17.

[11] Chen, S., H., Chen, M., C., & Liou, Y., C. (2014). Artificial chromosomes with genetic algorithm 2 (ACGA2) for single machine scheduling problems with sequence-dependent setup times. *Applied Softcomputing, 17*, 167-175.

[12] Abiyev, R., H., & Menekay, M. (2007). Fuzzy Portfolio Selection Using Genetic Algorithm. Soft Computing- A Fusion of Foundations, *Methodologies and Applications, Springer, Berlin/ Heidelberg*,*11*(12), 1157-1163.

[13] Abiyev, R. & Tunay, M. (2015). Optimization of High Dimensional Functions through Hypercube Evaluation. Computational Intelligence and Neuroscience, Volume 2015, 2015.

[14] Eduardo, L., R., José Luis, C., V., Belén, M., B., & Marcos Moreno-Vega J. (2014). Biased random key genetic algorithm for the tactical berth allocation problem. *Applied Softcomputing, 22*, 60-76.

[15] Vaisakh, K., Srinivas, L., R., & Meah, K. (2014). Genetic evolving ant direction particle swarm optimization algorithm for optimal power flow with non-smooth cost functions and statistical analysis. *Applied Softcomputing*, *13*, 4579-4593.

[16] Sörensen, K. (2013). Metaheuristics—the metaphor exposed. *International Transactions in Operational Research*,*22*(1), 3–18.

[17] Hansen, N., & Ostermeier, A. (2001). Completely derandomized self-adaptation in evolution strategies. *Evolutionary computation*, *9*(2), 159-195.

[18] Su, M-C., Su, S-Y., & Zhao, Y-X. (2009). A swarm-inspired projection algorithm. *Pattern Recognition*, *42*, 2764-2786.

[19] Belal, M., A., & Haggag, M., H. (2013). A structured-population genetic-algorithm based on hierarchical hypercube of genes expressions. *International Journal of Computer Applications*, *64*(22), 5-18.

[20] Chen, C., C. (2011). Two-layer particle swarm optimization for unconstrained optimization problems. *Applied Soft Computing*, *11*(1), 295-304.

[21] Precup, R., E., David, R., C., Petriu, E., M., Preitl, S., & Radac, M., B. (2012). Fuzzy control systems with reduced parametric sensitivity based on simulated annealing. *IEEE Transactions on Industrial Electronics*, *59*(8), 3049-3061.

[22] Wang, Y., Li, B., & Weise, T. (2013). Two-stage ensemble memetic algorithm: Function optimization and digital IIR filter design. *Information Sciences*, *220*, 408-424.

[23] Yazdani, D., Nasiri B., Azizi R., Sepas-Moghaddam, A., & Meybodi, M., R. (2013). Optimization in dynamic environments utilizing a novel method based on particle swarm optimization. *International Journal of Artificial Intelligence*, *11*(A13), 170-192.

[24] Weise, T., Skubch, H., Zapf, M., and Geihs, K. (2008). Global optimization algorithms and their application to distributed systems, Fachbereich 16: Elektrotechnik/ Informatik, Univ. Kassel.

[25] Holland, J. (1992). Adaptation in natural and artificial systems. Cambridge: University of

Michigan Press, Extended new Edition, MIT Press.

[26] Li, X., Yao, X. (2012). Cooperatively Coevolving Particle Swarms for Large Scale Optimization. *IEEE Transaction on Evolutionary Computation, 16*(2), 210-224.

[27] Liang, J., J., Qin, A., K., Suganthan, P., N., & Baskar, S. (2006). Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. *IEEE Transaction on Evolutionary Computation, 10*(3), 281-295.

[28] Zhang, J., & Sanderson, A., C. (2007). JADE: Self-Adaptive Differential Evoluation with Fast and Reliable Convergence Performance. *In Proceedings of theIEEE Congress on Evolutionary Computation* (pp. 2251-2258). Singapore.

[29] Bose, D., Biswas, S., Vasilakos, A., V., & Laha, S. (2014). Optimal filter design using an improved artificial bee colony algorithm. *Information Sciences 281*, 443–461.

[30] Brest, J., Greiner, S., Boskovic, B., Mernik, M., & Zumer, V. (2006). Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. *In IEEE Transactions On Evolutionary Computation, 10*(6), 646-657.

[31] Davison, V. E., and E. G. Sullivan. 1963. Mourning doves' selection of foods. J. Wildl. Manage. 27:373-383.

[32] Attracting Doves to Your Land. http://www.caes.uga.edu/extension/taylor/anr/documents/AttractingDovestoYourLand.pdf

[33] Hansen, N., Auger, A., Ros, R., Finck, S., & Pošík, P. (2010). Comparing results of 31 algorithms from the black-box optimization benchmarking BBOB-2009. *In Proceedings of the 12th annual conference companion on Genetic and evolutionary computation* (pp. 1689-1696). Prague: Czech Technical University.

[34] Grosan,C. (2009). A. Abraham, A Novel Global Optimization Technique for High Dimensional Functions. *Inter. Journal of Intelligent Systems 24*, 421–440.

[35] Dixon, L., C., W. and Szegö, G. (1978). "The global optimization problem: An introduction," in *Proc. Toward Global Optimization 2*, Amsterdam, Netherlands: North-Holland, pp. 1–15.

[36] Liang, J., J., Qu, B-Y., & Suganthan P., N. (2013). Problem Definitions and Evaluation Criteria for the CEC 2014 Special Session and Competition on Single Objective Real-Parameter Numerical Optimization. *Technical Report 201311, Computational Intelligence Laboratory*, Zhengzhou: Zhengzhou University, and *Technical Report*. Singapore: Nanyang Technological University, China.

[37] Erlich, I., Rueda, J., L., Wildenhues, S., & Shewarega, F. (2014). Evaluating the Mean-Variance Mapping Optimization on the IEEE-CEC 2014 Test Suite. *In Proceedings of the IEEE Congress on Evolutionary Computation* (pp. 1625-1632). Beijing, China.

[38] Chen, L., Zheng, Z., Liu, H., L., Xie, Shengli. (2014). An Evolutionary Algorithm Based on Covariance Matrix Leaning and Searching Preference for Solving CEC 2014 Benchmark Problems. *In Proceedings of theIEEE Congress on Evolutionary Computation* (pp. 2672-2677). Beijing, China.

[39] Mallipeddi, R., Wu, G., Lee M., & Suganthan, P., N. (2014). Gaussian Adaptation based Parameter Adaptation for Differential Evolution. *In Proceedings of the IEEE Congress on Evolutionary Computation* (pp. 1760-1767). Beijing, China.

[40] Yashesh D., Deb K., and Bandaru, S. (2014). Non-Uniform Mapping in Real-Coded Genetic Algorithms. *In Proceedings of the IEEE Congress on Evolutionary Computation* (pp. 2237-2244). Beijing, China.

[41] Bujok, P., Tvrdık, J., & Polakova, R. (2014). Differential Evolution with Rotation-Invariant Mutation and Competing-Strategies Adaptation. *In Proceedings of the IEEE Congress on Evolutionary Computation* (pp. 2253-2258). Beijing, China.

[42] Elsayed, S., M., Sarker, R., A., Essam D., L., & Hamza N., M. (2014). Testing United Multi-Operator Evolutionary Algorithms on the CEC2014 Real-Parameter Numerical Optimization. *In Proceedings of the IEEE Congress on Evolutionary Computation* (pp. 1650-1657). Beijing, China.

[43] Tanabe, R., & Fukunaga, A., S. (2014). Improving the Search Performance of SHADE Using Linear Population Size Reduction. *In Proceedings of theIEEE Congress on Evolutionary Computation* (pp. 1658-1665). Beijing, China.

[44] Qu, B., Y., Liang, J., J., Xiao, J., M., & Shang, Z., G. (2014). Memetic Differential Evolution Based on Fitness Euclidean-Distance Ratio. *In Proceedings of the IEEE Congress on Evolutionary Computation* (pp. 2266-2273). Beijing, China.

[45] Hu Z., Bao Y., and Xiong T. (2014) Partial Opposition-Based Adaptive Differential Evolution Algorithms: Evaluation on the CEC 2014 Benchmark Set for Real-parameter Optimization. . *In Proceedings of the IEEE Congress on Evolutionary Computation* (pp. 2259-2265). Beijing, China.

[46] Li, Z., Shang, Z., Qu, B., Y., & Liang J., J. (2014). Differential Evolution Strategy based on the Constraint of Fitness Values. *In Proceedings of the IEEE Congress on Evolutionary Computation* (pp. 1454-1460). Beijing, China.

[47] Polakov, R., Tvrdık J., Bujok, P. (2014). Controlled Restart in Differential Evolution Applied to CEC2014 Benchmark Functions. *In Proceedings of the IEEE Congress on Evolutionary Computation* (pp. 2230-2236). Beijing, China.

[48] Dourado Maia R., Nunes de Castro L., and Matos Caminhas, W. (2014). Real-Parameter Optimization with OptBees. *In Proceedings of the IEEE Congress on Evolutionary Computation* (pp. 2649-2655). Beijing, China.

[49] Xu, C., Huang, H., & Ye, S. (2014). A Differential Evolution with Replacement Strategy for Real-Parameter Numerical Optimization. *In Proceedings of theIEEE Congress on Evolutionary Computation* (pp. 1617-1624). Beijing, China.

[50] Burke, E., K., Elliman, D., G., & Weare, R., F. (1995). A Hybrid Genetic Algorithm for Highly Constrained Timetabling Problems. *In Proceedings of the 6th International Conference on Genetic Algorithms* (pp. 15-19). Pittsburgh, USA.

[51] Burke, E., K., Newall, J., P. (1999). A multistage evolutionary algorithm for the timetable problem. IEEE Transaction on Evolutionary Computation 3(1), 1085-1092.