A New Approach on MOPSO Using Simulated Annealing & Genetic Algorithm Concepts

Subodh Gupta[1], Anand Jawdekar[2]
[1, 2] Department of Computer Science Engg
[1, 2] SRCEMBanmore, India

Abstract-*Optimization plays an important role in many areas of science, management, economics and engineering. This paper presents a new Multi-Objective Particle Swarm Optimization (MOPSO) algorithm which has been optimized using genetic algorithm that has new components: selection using genetic algorithm, crossover and improved PSO. The genetic algorithm optimizes the results by applying various operators of genetic. The selection process is also different which somehow improves the results. Besides, crossover and mutation adopted in the proposed algorithm also contributes to the betterment of the results. The performance of the proposed Simulated Annealing based MOPSO algorithm was compared with four popular multi-objective algorithms in solving standard test functions. Their performance measures were mainly calculated on hypervolume and proposed algorithm was generally better from previous.*

Keywords-PSO, Genetic Algorithm, MOPSO, SA; RWS.

I. INTRODUCTION

An optimization issues can be defined as the problem of discovering the best solution among the optimal solutions. In this, Particle swarm optimization (PSO) is a heuristic global optimization method, proposed originally by Kennedy and Eberhart in 1995. It is now one of the most commonly used optimization techniques. This survey presented a comprehensive investigation of PSO. On one hand, we provided advances with PSO, including its modifications, hybridization (with genetic algorithm, simulated annealing, Tabu search, artificial immune system, ant colony algorithm, artificial bee colony, differential evolution, harmonic search, and biogeography-based optimization), extensions (to multiobjective, constrained, discrete, and binary optimization), theoretical analysis (parameter selection and tuning, and convergence analysis), and parallel implementation (in multicore, multiprocessor, GPU, and cloud computing forms) [1], where an intelligent agent represents a system that perceives its environment and takes action that maximizes its success chance. Currently popular approaches of AI include traditional statistical methods [2], traditional symbolic AI, and computational intelligence (CI) [3]. CI is a fairly new research area. It is a set of nature-inspired computational methodologies and approaches to address complex real-world problems to which traditional approaches are ineffective or infeasible. CI includes artificial neural network (ANN), fuzzy logic, and evolutionary computation (EC) [4].

In order to apply the PSO strategy for solving multi-objective optimization problems, it is obvious that the original scheme has to be modified. The solution set of a problem with multiple objectives does not consist of a single solution (as in global optimization). Instead, in multi-objective optimization, we aim to find a set of different solutions (the so-called Pareto optimal set). In general, when solving a multi-objective problem, three are the main goals to achieve:

1. Maximize the number of elements of the Pareto optimal set found.
2. Minimize the distance of the Pareto front produced by our algorithm with respect to the true (global) Pareto front (assuming we know its location).
3. Maximize the spread of solutions found, so that we can have a distribution of vectors as smooth and uniform as possible.

Real-life engineering optimization problems need Multiobjective Optimization (MOO) tools. These problems are highly nonlinear. As the process of Multiple Criteria Decision-Making (MCDM) is much expanded most MOO problems in different disciplines can be classified on the basis of it. Thus MCDM methods have gained wide popularity in different sciences and applications [5]. Meanwhile the increasing number of involved components, variables, parameters, constraints and objectives in the process, has made the process very complicated. However the new generation of MOO tools has made the optimization process more automated, but still initializing the process and setting the initial value of simulation tools and also identifying the effective input variables and objectives in order to reach the smaller design space are still complicated. In this situation adding a preprocessing step into the MCDM procedure could make a huge difference in terms of organizing the input variables according to their effects on the optimization objectives of the system [6].

II. SIMULATED ANNEALING (SA)

SA is probably stated [6] to be the oldest among the many Meta heuristics and without doubt some of the first algorithms that had a specific approach to flee from local minima. The development of the algorithm are in statistical mechanics (town algorithm) and it used to be first furnished as a search algorithm for CO disorders in Kirkpatrick et al. and Cerny. The foremost concept is that making improvements to candidate solutions are at all times accepted while non-making improvements to solutions are approved with a targeted chance. The probability of accepting non-bettering solutions is calculated in step with the current temperature of the algorithm. This method is analogous to the annealing system of metals and glass, which anticipate a low energy configuration when cooled with a proper cooling time table. Irrespective of search method, algorithm combine two methods: Random walk and iterative improvement.

The algorithm begins with the aim of producing a preliminary resolution (both randomly and heuristically developed) and with an excessive preliminary temperature, T, which corresponds to a high chance of accepting non-bettering options. The temperature is gradually reduced as the find progresses in order that the chance of accepting non-making improvements to solutions can also be lowered. At temperature zero, T0, the algorithm operates like an improving heuristic, i.e., only improving solutions are accepted.

III. GENETIC ALGORITHM

Genetic algorithms (GA) is an adaptive heuristic search algorithm based on the evolutionary ideas of natural selection and genetics. As such, they represent an intelligent exploitation of a random search used to solve optimization problems. The basic techniques of GA are designed to simulate processes in natural systems necessary for evolution, especially those following the principles first laid down by Charles Darwin of "survival of the fittest". GA is based on an analogy with the genetic structure and behavior of chromosomes within a population of individuals.

The GA maintains a population of chromosomes (solutions) associated with fitness values. Parents are selected to mate on the basis of their fitness, producing offspring via a reproductive plan. Consequently, highly fit solutions are given more opportunities to reproduce, so that offspring inherit characteristics from each parent. Once an initial population is randomly generated, the algorithm evolves through operators:

- A selection which equates to the survival of the fittest;
- Crossover which represents mating between individuals;
- Mutation which introduces random modifications.

Using a selection operator alone will tend to fill the population with copies of the best individual from the population. Using selection and crossover operators will tend to cause the algorithms to converge to a good, but sub-optimal, solution. Using mutation alone induces a random walk through the search space, while using selection and mutation creates a parallel, noise-tolerant, hill-climbing algorithm.

GAs are quite popular and applicable to many domains like industrial design, scheduling, network design, routing, time series prediction, database mining, control systems, artificial life systems, as well as in many fields of science [7]. An individual in the population can be selected more than once, with all individuals in the population having a chance of being selected to reproduce into the next generation. There are several schemes for the selection process: roulette wheel selection and its extensions, scaling techniques, tournament, elitist models, and ranking methods [8, 9].

IV. ROULETTE WHEEL SELECTION (RWS)

This is the simplest selection method. In this method all chromosomes (individuals) population are placed on RW in a manner of fitness value [10]. On RW, each individual is allotting a segment. In RW each segment size is proportional to fitness value of individual, the larger the value, bigger the segment is. Then wheel is virtually spin. A simple scenario of selection strategy in Roulette Wheel mechanism in fig 1.5. The individual is comparing to the segment on which RW stops are then chosen. The steps are repeated until the best individual is chosen. Individuals with maximum fitness have more probability of selection. This may lead to biased selection towards high fitness individuals. It can also possibly miss the best individuals of a population. There is no confirmation of which good individuals will find a place in next generation.

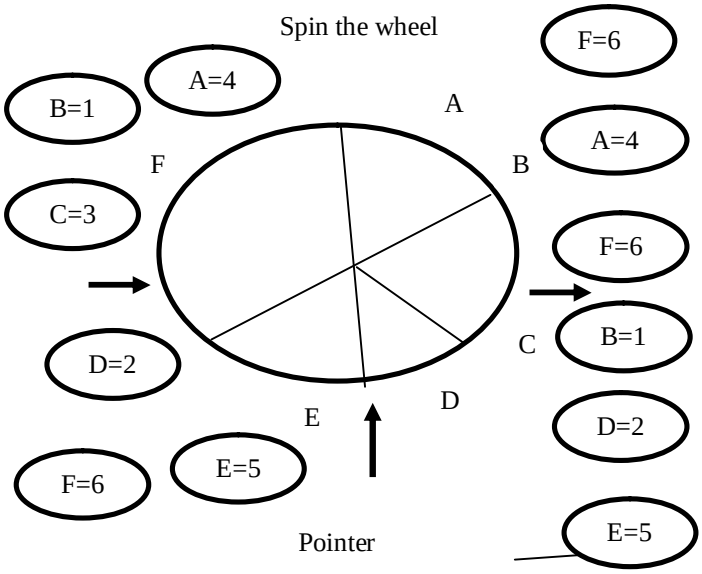


Fig. 1. Selection strategy with roulette wheel mechanism

In this method selection is done using exploitation technique in its approach. The usual population fitness for ith generation in RWS is computed as:

$$\overline{FRW_{i,j}} = \frac{\sum_{j=1}^{N} FRW_j}{N}$$

Where i varies from 1 to ngen and j varies from 1 to N. Therefore, the probability for selecting the jth string is

$$\overline{PRW_{i,j}} = \frac{\sum_{j=1}^{N} FRW_j}{N}$$

Where N is the population size and FRWj is the fitness of individual j.

V. LITERATURE REVIEW

Man-Fai et al. [11] presents a new MOPSO algorithm that has two new components: leader selection and crossover. The proposed algorithm leader selection, Space Expanding Strategy (SES) that guides moving particles towards boundary of objective space in every generation so expand the objective space rapidly. To improve the convergence used of crossover to maintain the stability of the generated solutions (exploitation). The proposed solution MOPSO algorithm performance was compared with three popular multi-objective algorithms in solving fifteen standard test functions.

D. Cai et al. [12] proposed a new multi-purpose PSO algorithm established on decomposition of the target house (MPSO/D) for fixing MOO problems. An update strategy situated on decomposition is proposed to make each sub-vicinity in the goal house have a Pareto superior answer. This replace method can maintain really well the multiplicity of the obtained solutions, and the diversity is fundamental for fixing MOPs. Concurrently, to toughen the convergence of MPSO/D, the crowding distance is used to calculate the health value of these options for resolution operators. Additionally, MPSO/D compares with NSGAII, MOEA/D and NNIA on sixteen test occasions with intricate PS or many local PFs, simulation outcome verify that MPSO/D vastly outperforms MOEA/D, NSGAII and NNIA on most test issues.

R. A. Zein Eldin [13] presented a hybrid approach founded o scatter search and simulated annealing to solve the MOO. The performance of algorithm measure over different test problem with other approaches, and proposed approach is effective and competitive with the other developed approaches.

Ulungu et al. [14] investigate another multi-objective simulated annealing method. In this method, the acceptance criteria is again calculated using a weighted sum of the objectives and an archive set of possibly optimal solutions is maintained. While this method is very similar to Serafini's prior work and results are presented on a formulation of the knapsack problem, no comparison is made to either of the previously discussed algorithms so its relative performance is difficult to determine.

Kennedy and Eberhart [15] initially proposed the swarm strategy for optimization. Particle swarm optimization (PSO) is a stochastic optimization technique that draws inspiration from the behavior of a flock of birds or the collective intelligence of a group of social insects with limited individual capabilities. In PSO, individuals, referred to as particles, are "flown" through hyper dimensional search space PSO seems particularly suitable for multiobjective optimization mainly because of the high speed of convergence that the algorithm presents for single objective optimization.

VI. PROPOSED WORK

In this work, we proposed a new algorithm to improve the method of previous work. For this we optimized the method using MOPSO and then after apply GA and Simulated Annealing. In the beginning of proposed model a set of swarm population is taken, whose velocity, number of iteration counter and inertia weight is initialized. After the initialization process, compute the bi-objective function is a NDS. In every generation, select a leader on fitness value to find its DS or NDS over local or global set. If the solution is dominated store such in external archive to other from non-dominated one.

The particle fitness value is evaluating using roulette wheel selection method. The local and global best position of particle is selected, from members distance between non-dominated local set and global set are measured in objective space. The position of local and global members is evaluated. Update the particle velocity and position update using local and global set. Evaluate bi-objective function over the test problem i.e. (ZDT, ZDT2 and soon). The values of C1 (personal learning coefficient) and C2 (global learning coefficient) were assigned randomly between 2, inertia weight is varying between 0.8-1.0.

The updated position of objective space is added to local and global non-dominated set to expand and update if in between min or max of problem. The external set is update with members of global set external Pareto optimal. If the external archive is more populated with particle of DS, so reduce them and search non-dominated external Pareto. Apply GA cross over operator to compare new best values from previous values. The current fitness is set as pbest, if the particle best position value is improved over pbest. The optimal single extreme value is selected as gbest, update lbest and gbest for each particle. Compute average fitness of particle and also evaluate the mutation over which new mutation solution is obtained. The new solution is accepted when the no. of iterations not exceeds if so then update inertia weight using simulated annealing and follow process and find one solution to the solution producer and stop.

Proposed Algorithm

Step 1:- Initialize swarm population, maximum iteration and inertia weight

Step 2:- Fitness evaluation and Pareto dominance for ranking particles (solutions)

Step 3:- Store memory: Personal Best (Pbest) = swarm population (and their ranks)

Step 4:- Store non-dominated solutions in External Archive

Step 5:- Find the acceptance criterion using Boltzmann probability equation:

$$prob(p) = exp\frac{-\Delta E}{K_b T}$$

Where in this expression,K_b is a constant, ΔE is improvement in energy value from one point to the next, T is the temperature, it is used to control variable that controls the flow of annealing. The acceptance criterion is commonly called as the Metropolis criterion. And the improving and deteriorating move of this acceptance criterion has been proposed by Galuber:

$$prob(p) = \frac{\exp(-\Delta E / K_b T)}{sum(\exp(-\Delta E / K_b T))}$$

Step 6:- Select cell index with $prob(p)$using roulette wheel method:

Step 7:- Find the global leader g on the basis of repository and selected unique member of cell.

Step 8:- Determine linear decreasing inertia weight using this expression:

$$w = w_{max} - ((w_{max} - w_{min}) \times i)/it$$

Where$w_{max} = 0.8$,$w_{min} = 0.1$ I tends to number of population size, it tends to maximum no. of iteration

Step 9:- Determine the velocity (V) and position(x) of each particle. It will be changed by the pbest and gbest. Find the gbest using V and x updating of the particle is shown in the below equation:

$$V_{mn}(t+1) = w \times V_{mn}(t) + c_1 \times r_1 (pB_{mn}(t) - x_{mn}(t)) + c_2 \times r_2 (gB_{mn}(t) - x_{mn}(t))$$

$$x_{mn}(t+1) = x_{mn}(t) + V_{mn}(t+1)$$

Where w stands for inertia weight which is used to control the effect of previous values of velocities, $V_{mn}(t+1)$ velocities of particle m at iterations n, positions of particle m^{th}at iterations n^{th}

Step 10:- Find the min and max value of position

Step 11:-For each particle compute fitness function$F_m(k)$ and the average fitness $F_{avgm}(k)$

Step 12:-

The crossover operator is used to compare new best values from previous values. If the particle fitness is superior than *Pbest*, the current fitness is set as *Pbest* .The optimal individual extreme value is selected as *Gbest.*

Step 13:- Update the position of the particles

Step14:-Evaluate the best decision of each particle $F_m(k+1)$ and the average fitness

$$F_{avgm}(k+1)$$

Step 15:- Calculate the percentage of mutation using this expression:

$$pm = \left(\frac{1-(it-1)}{MaxIt-1}\right)^{\frac{1}{mu}}$$

Where mu is mutation rate, MaxIt is maximum iteration and pm is percentage of mutation

Step 16:- The mutation operator is used to obtain the new mutation.

Step 17:- Find whether to accept a new solution.

Step 18:- Calculate the caused change amount of fitness value between two positions

$\Delta F = F_m(k+1) - F_{avgm}(k+1)$

If $\Delta F < 0$, a new decision vector is accepted. Otherwise it will remain the old solution.

Step 19:- If the fitness of fitted individual in F_m is not enough then return to step 14

Step 20:- It the repository is full then Delete selection process and stop condition with non-dominated members in repository.
Step 21:- Obtain the final optimal solution.

VII. RESULT ANALYSIS

The results have been calculated for best 100 populations for finding the hypervolume. The repository members for each iteration are changed and keeps on increasing until they reach to the maximum population size of 100 individuals. The individuals always reach to 100 in a limited number of iterations. To illustrate the performance of the new proposed algorithms, ZDT and MOP algorithms were used for performance comparisons in few important benchmark MOPs. The comparison between the base and the proposed algorithm have been performed. The proposed algorithm uses decomposition to select swarm leaders and update the external archive based on simulated annealing. The external archive stores the solutions with the best aggregation values. This algorithm uses simulated annealing based multi-objective PSO based to optimize the results.

Bi-objective test problems are adopted in performance investigation. ZDT1, ZDT2, ZDT3, MOP2. The results are calculated on these functions. Three popular performance measure used is Hypervolume (IHV) [15]. When IHV is higher, the convergence and diversity of the found solutions is better.

Fig 2 Final GUI of Proposed Result

Table1: IHV Comparison between Base and Proposed System on 100 Iteration

Test Problems	Iterations	Base results Hypervolume	Proposed results Hypervolume
ZDT1	100	0.360000	0.570000
ZDT2	100	0.100000	0.290000
ZDT3	100	0.380000	0.690000
MOP2	100	0.030000	0.130000

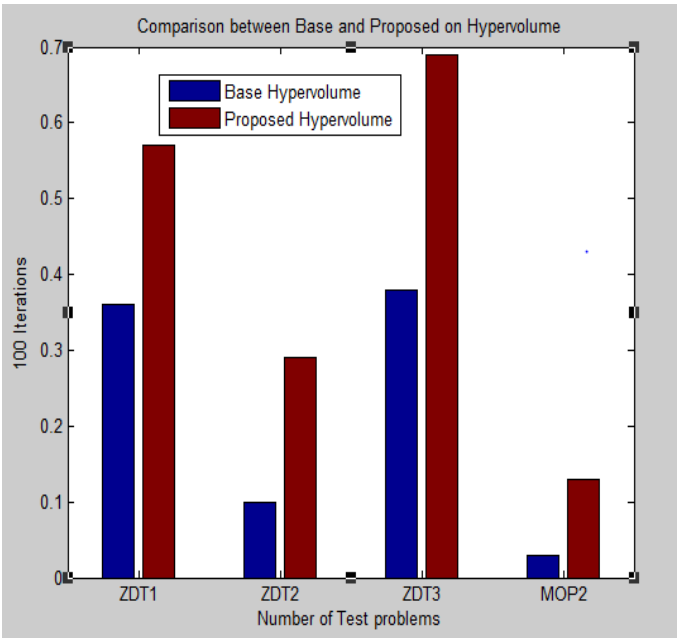


Fig.3. Comparison between base and Proposed IHV on 100iteration for different test problems

Table2 IHV Comparison between Base and Proposed System on 150 Iteration

Test Problems	Iterations	Base results Hypervolume	Proposed results Hypervolume
ZDT1	150	0.130000	0.530000
ZDT2	150	0.240000	0.280000
ZDT3	150	0.050000	0.560000
MOP2	150	0.010000	0.130000

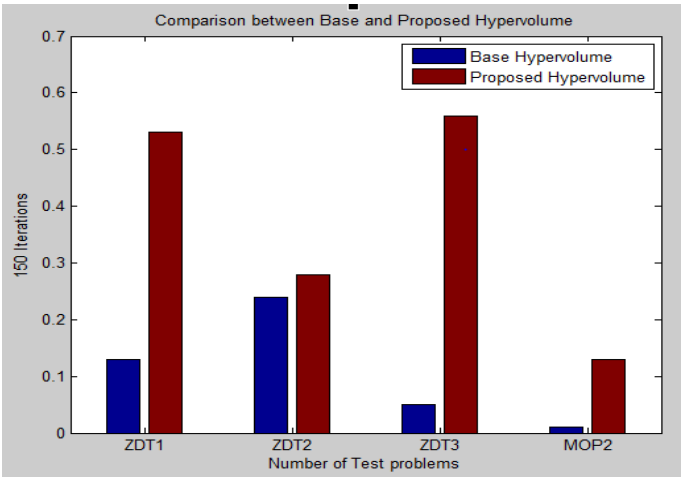


Fig 4 Comparison between base and Proposed IHV on 150 iteration for different test problems

Table3: IHV Comparison between Base and Proposed System on 200 Iteration

Test Problems	Iterations	Base results Hypervolume	Proposed results Hypervolume
ZDT1	200	0.450000	0.570000
ZDT2	200	0.160000	0.310000

ZDT3	200	0.100000	0.700000
MOP2	200	0.020000	0.150000

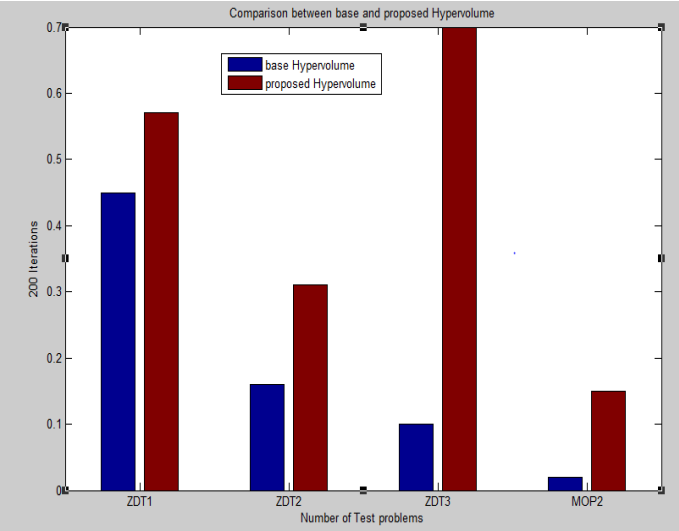


Fig 5 Comparison between base and Proposed IHV on 200 iteration for different test problems

Table4: IHV Comparison between Base and Proposed System on 250 iteration

Test Problems	**Iterations**	**Base results Hypervolume**	**Proposed results Hypervolume**
ZDT1	250	0.400000	0.570000
ZDT2	250	0.070000	0.310000
ZDT3	250	0.090000	0.650000
MOP2	250	0.010000	0.150000

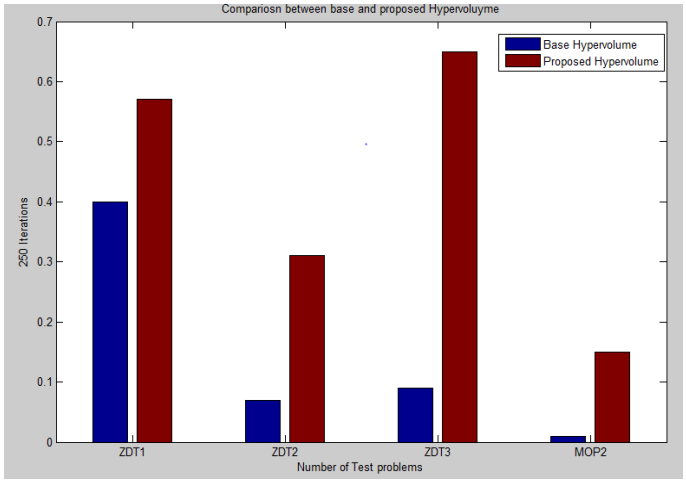


Fig 6 Comparison between base and Proposed IHV on 250 iteration for different test problems

Table5 IHV Comparison between Base and Proposed System on 300 iteration

Test Problems	**Iterations**	**Base results Hypervolume**	**Proposed results**
ZDT1	300	0.120000	0.660000
ZDT2	300	0.130000	0.330000
ZDT3	300	0.020000	0.730000
MOP2	300	0.020000	0.120000

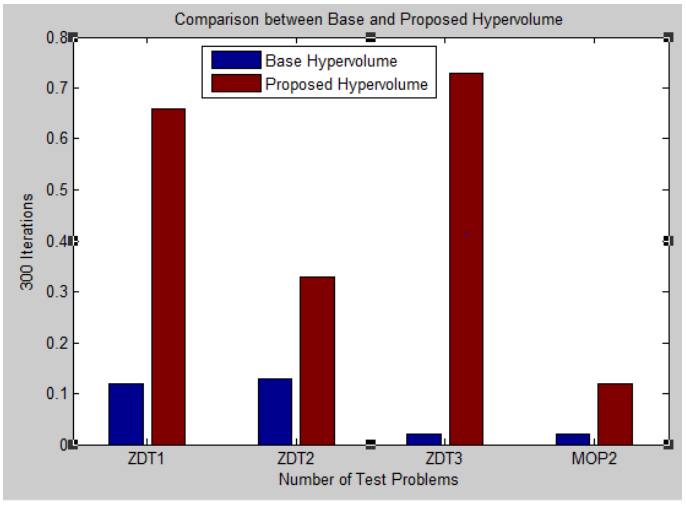
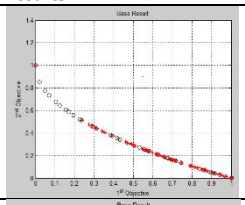
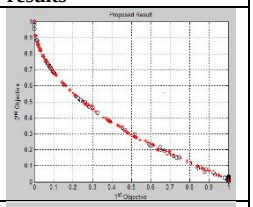
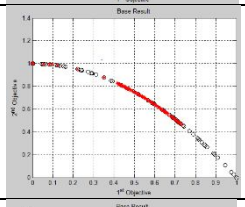
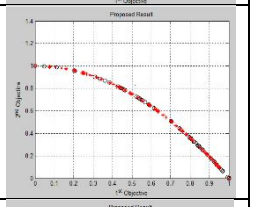
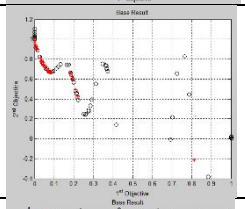
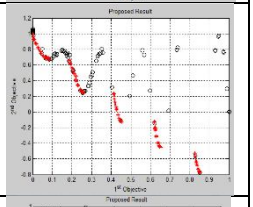
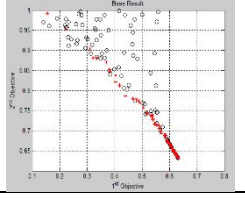
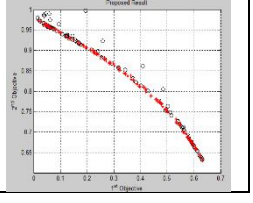


Fig 7 Comparison between base and Proposed IHV on 300 iteration for different test problems

Table 6 : Pareto Front Comparison between Base and Proposed System on 100 Iteration

Test Problems	**Base Pareto results**	**Proposed Pareto results**
ZDT1 (100)		
ZDT2 (100)		
ZDT3 (100)		
MOP2 (100)		

VIII. CONCLUSION

Today optimization is gaining much more attention of researchers due to ability of solving global best solution for

real life problems. In this work, presents a new approach based on MOPSO using Simulated Annealing (SA) and GA to optimize the inherited algorithm that has new components: choice by inherited algorithm, crossover with mutation and improved PSO. The inherent algorithm optimize the results by applying various operators of genetic. The selection process is also different which somehow improves the results. The proposed MOPSO based SA & GA algorithm was compared with four popular multi-objective algorithms in solving average test functions namely ZDT, ZDT2, ZDT3 and MOP2. The result performance evaluate on test function and measure on hyper volume that shows better with proposed solution than previous one. This update strategy can preserve fairly well the variety of the obtained solutions, and the diversity is essential for solving MOPSO. Concurrently, to get better the convergence of MOPSO, the swarm space is used to analyze the fitness worth of those solutions for choice operators. In future, proposed solution can be tested on realistic data to authenticate in different benchmark problem.

REFERENCES

[1] Schaeffer, Jonathan. One Jump Ahead: Challenging Human Supremacy in Checkers ,2009, Springer, ISBN 978-0-387-76575-4.

[2] McCorduck, Pamela (2004), Machines Who Think (2nd ed.), Natick, MA: A. K. Peters, Ltd., ISBN 1-56881-205-1, p. 480-483.

[3] Yannakakis, Geogios N (2012). "Game AI revisited". Proceedings of the 9th conference on Computing Frontiers: 285–292.

[4] Zimmermann, H.-J., Fuzzy Set Theory and Its Applications, Kluwer Academic Publishers, Second Edition, Boston, MA, 1991.

[5] Kuhn, H.W. and A.W. Tucker, "Nonlinear Programming," Proc. 2nd Berkeley Symp. Math. Stat. Prob., 481-492, 1951

[6] B Suman1 and P Kumar, “A survey of simulated annealing as a tool for single and multiobjective optimization”, Journal of the Operational Research Society, pp.1143–1160, 2006.

[7] Bies R., M. Muldoon, B. Pollock, S. Manuck, G. Smith, M. Sale, A Genetic Algorithm-based, Hybrid Machine Learning Approach to Model Selection, Journal of Pharmacokinetics and Pharmacodynamics, 2006, 196–221.

[8] Chipperfield A., P. J. Fleming, H. Pohlheim, C. M. Fonseca, Genetic Algorithm Toolbox for Use with MATLAB, 1993.

[9] Davis L., Handbook of Genetic Algorithms, Van Nostrand Reinhold, 1991.

[10] Rakesh Kumar, Jyotishree, “Blending Roulette Wheel Selection & Rank Selection in Genetic Algorithms”, International Journal of Machine Learning and Computing, Vol. 2, No. 4, August 2012.

[11] Man-Fai Leung, Sin-Chun Ng, Chi-Chung Cheung and Andrew K Lui, “A New Algorithm based on PSO for Multi-objective Optimization”, IEEE 2015.

[12] D. Cai, W. Yuping and Y. Miao, “A new multi-objective particle swarm optimization algorithm based on decomposition”, Elsevier Information Science, 2015.

[13] R. A. ZeinEldin, “A Hybrid SS-SA Approach for Solving Multi-Objective Optimization Problems”, European Journal of Scientific Research, vol. 121, no. 3, pp. 310-320, 2014.

[14] E.L. Ulungu, J. Teghaem, Ph. Fortemps, and D. Tuyttens. MOSA method: a tool for solving multiobjective combinatorial decision problems. Journal of Multi-criteria Decision Analysis, 8: 221–236, 1999.

[15] Kennedy and R. C. Eberhart, Swarm Intelligence. San Mateo, CA: Morgan Kaufmann, 2001.