Abstract:

Evolutionary algorithms normally produce better results when they are applied on optimization problems. This report provides a detailed description of a software that applies evolutionary algorithms to find the optimum solution to any mathematical function. The software uses two evolutionary algorithms i.e. genetic algorithm and particle swarm optimization algorithm. The software has a graphical user interface that allows the user to enter any mathematical function to optimize and select different parameters of evolutionary algorithms. The software produces all the results graphically. Additionally, the software also allows the user to export all the results to an excel file.

This report is divided into five main sections. The first section is literature overview of different evolutionary algorithms and their parameters. The second section introduces the software and its functionality. The third section provides a detailed description of the rationale behind developing this software. The fourth section discusses the evolutionary algorithms that the software uses to optimize any mathematical function. The fifth and last section introduces the graphical user interface of the software.

1. Literature Overview:

Evolutionary Algorithms imitate the behaviour of natural processes to find the solution to real world problems. They are well suited for optimization problems involving multiple conflicting objectives [1]. These algorithms usually outperform traditional algorithms when they are applied to multi-objective complex optimization problems. However, these algorithms are very sensitive and a small change to input parameters that drive these algorithms can have a significant effect on the output produced by these algorithms. Parameter control is one of the promising areas of research in evolutionary algorithms. [2] Provides a classification of various parameter control mechanisms.

There are a number of different types of evolutionary algorithms, including genetic algorithms (GA), genetic programming, evolutionary programming, differential evolution, ant colony optimization and particle swarm optimization etc. Different evolutionary algorithms are suited for different optimization problems. [3] Compares the results produced by differential evolution, particle swarm optimization and other evolutionary algorithms on numerical problems. According to the results in [3], differential evolution outperforms other evolutionary algorithms. However, this is not true in call cases. As previously mentioned, these algorithms are very sensitive to the input parameters and the problems that they are applied to. Differential evolution maybe more suitable for some numerical problems, similarly genetic algorithms can be more suitable for other numerical problems.

In genetic algorithms, a population of candidate solution is evolved towards a better solution. The process is repeated over a certain number of generations or iterations. Population consist of individuals or chromosomes and they are evolved using biological processes like crossover and mutation. It then uses the process of natural selection to pass the fit individuals to the next generation and discard the individuals that are not fit. Transforming a real world problem into a problem that could be solved by using genetic algorithm is called representation. There are various chromosome or individual

representation techniques such as binary, permutation and floating point. [4] Applies genetic algorithm techniques to floating point individual representation.

There are various selection methods that are used to select and discard individuals for future generations. Some of the common selection methods are Tournament Selection, Roulette Wheel Selection, Rank selection and Truncation Selection. [5] Compares different selection methods for travel salesman problem. [6] Compares different selection methods and crossover operations. [7] Discusses the performance of Tournament selection, Fitness Proportionate Selection and Rank Selection on Travel Salesman problem. [8] Proposes a hybrid selection scheme by blending roulette wheel and rank selection schemes and argues that it increases the performance of genetic algorithm. Again, it's very difficult to generalize the findings of all these papers and declare a selection method that is best. It all depends on the optimization problem, representation of individuals and reproduction strategies.

Process of reproduction produces offspring from the parent population. Reproduction involves the process of recombination or crossover and mutation. Recombination transfer the genes between parents to produce offspring. There are various recombination types in genetic algorithms including one point crossover, arithmetic single crossover, arithmetic simple crossover and arithmetic whole crossover. [9] Studies different crossover operators that are applied on job shop scheduling problem. [10] Discusses a hybrid adaptive crossover method called the selective crossover and argues that it provided better results for large set of optimization problems. [11] Discusses another crossover technique known as ring crossover. [12] Discusses a crossover method called the moon crossover and argues that it is well suited for travel salesman problem.

Mutation alters the value of one or more genes in an individual. It introduces new material into the population. It explores new solution areas and allows an algorithm no to get stuck at the local minima or maxima. For real valued representation of individual, mutation alters a gene by a certain value or step

size. There are different distributions that are used to generate the mutation step size including normal distribution [13] and Cauchy distribution [14]. [15] Analyzes the performance of different mutation operators to solve the travel salesman problem. [16] Introduces a hybrid mutation technique called the adaptive mutation and argues that it produces better results.

In Particle Swarm Optimization (PSO) algorithm, a candidate solution is evolved towards a better solution by repeating the process over a certain number of time steps or iterations. PSO evolved the candidate solution in a different way than GA. PSO is inspired by social behaviour of birds flocking. It consists of a swarm of particles. Each particle has a certain position and velocity. All the particles explore the search space for better solution. The movement of each particle is controlled by its own best position and the best position of a particle in the entire swarm. [17] and [18] gives detailed information about PSO algorithm.

Every particle updates its velocity according to a certain velocity update rule. There are a number of velocity update rules including Linear Increasing Inertia Velocity Update Rule, Linear Decreasing Inertia Velocity Update Rule and Constriction Coefficient Velocity Update Rule. [19] Discusses different velocity update rules in PSO algorithm and their experimental results on different problems. Velocity update is controlled by inertia weight, introduced by [20]. It is a mechanism that controls the contribution of previous velocity by adjusting inertia weight. [21] Discusses non-linear decreasing inertia weight. It states that it has shorter exploration time than linear inertia weight update rule.
Several evolutionary algorithms are discussed in this section. Different algorithms are suited for different problems. Similarly, parameters such as population size, selection methods in GA and velocity update rules in PSO affects the performance of these algorithms in a great deal. Same algorithm with one parameter settings can give better results than the same algorithm with other parameter settings. There is no best evolutionary algorithm that is suited for all the problems.

2. Introduction to the Software:

The software uses Genetic Algorithm (GA) and Particle Swarm Optimization Algorithm (PSO) to optimize any mathematical function. The user can input any mathematical function and the software would run the algorithm and produce the results. The user can view the data or the graphs using the graphical user interface of the software. The user can also import the data to an excel file. The software creates an excel file, transfer all the data to it and create charts in different sheets within the excel file. The user can change the parameter values of evolutionary algorithms and re-run the software.

The user can select a variety of genetic algorithm parameters and change their values. The user can change the following parameter values in genetic algorithms:

- Function lower and upper bound
- Population and offspring size
- Mating selection method i.e. tournament selection, fitness proportionate selection and rank selection
- Survivor selection method i.e. truncation selection
- Recombination type i.e. one point crossover, arithmetic single crossover, arithmetic simple crossover and arithmetic whole crossover
- Crossover rate
- Mutation type i.e. Normal distribution and Cauchy distribution
- Mutation rate
- Number of generations
- Number of runs

The user can select a number of parameters in particle swarm optimization algorithm including:

- Function lower and upper bound

- Particle velocity update rule i.e. linear decreasing inertia velocity update rule and linear increasing inertia velocity update rule
- Particle positive clamp velocity
- Particle negative clamp velocity
- Number of particles in a swarm
- Number of time steps for the algorithm to run
- Number of runs

The user has different options to view and analyze the results. The user can view the following data and plots:

- Best particle fitness in each time step (average of all runs)
- Average fitness of swarm in each time step (average of all runs)
- Particle dispersion in each time step (average of all runs)

The user can also import all the data and plots to an excel file. The excel file would contain the following data and plots:

- Best particle fitness in each time step in each run
- Average fitness of swarm in each time step in each run
- Particle dispersion in each time step in each run

3. Objectives for Developing the Software:

Evolutionary algorithms are very sensitive to the input parameters and a very small change in one of the parameter can change the outcome of the algorithms to a great degree. It is a very time consuming process to fine tune an algorithm for a particular optimization problem. For every small change in the algorithm the researcher has to go back to the code, change values in the code, re-compile and re-run the code, transfer the results to a file and analyze the results. This whole process takes a lot of time and rather than concentrating on the output, the researcher has to spend a lot of time re-compiling the code and exporting the data. Changing the input optimization function is even more time consuming. In that case the researcher has to change a large portion of the code.

This Software is designed and developed to solve this time consuming process by allowing the researcher to change all the parameter values using a user friendly graphical user interface. The researcher does not need to change the code at all. The researcher can simply change the parameter values graphically and the software would produce all the results in a graphical user interface. It saves a lot of time and energy of the researchers. The main aim of this software is to allow the researchers to spend more time in analyzing and fine tuning the algorithm for a particular optimization problem rather than wasting time on changing the code.

The software has a very user friendly interface. It allows the user to enter any simple or complex mathematical function for optimization. This software is ideal for researchers working on complex mathematical function to optimize. This software is also ideal for someone new to evolutionary algorithms. It gives a simple overall picture of the working of genetic algorithm and particle swarm optimization algorithm. The software would help someone new to these algorithms to familiarize himself/herself with all the concepts of these algorithms.

4. Evolutionary algorithms used by the Software:

This section briefly explains the evolutionary algorithms that the software uses to optimize a mathematical function. This section would briefly explain different concepts related to genetic algorithms and particle swarm optimization algorithms.

4. 1 Genetic Algorithms (GA):

In genetic algorithms a population called individuals, creatures or phenotype is evolved towards a better solution. This evolution process is composed of generations. With each successive generation, the population evolves towards a set of better solutions. Biological and evolutionary methods are applied on individuals in a population in each successive generation. Biological processes such as crossover and mutation are applied on individuals to generate offspring. Offspring produced are then selected and discarded based on evolutionary selection methods. Best offspring are passed on to the next generation and offspring that are not suitable are discarded. This process is repeated in each iteration or generation. In this way, the population evolves towards better and better solutions. The algorithm stops when a desired solution is found or after a certain number of predefined iterations.

The total number of individuals in a population remains the same in each generation. The software allows the user to enter the individual population size and the offspring population size. The algorithm then selects the offspring number of offspring from the individual population, applies the reproduction method (recombination and mutation) and produce the offspring. The algorithm then uses the truncation survivor selection method that selects the population size number of best individuals from both the parent and offspring population.

For example if the population size is 20 and the offspring population size is 10, then the algorithm would select 10 individuals from the population based on a selected selection method. The algorithm would then produce offspring from these selected individuals. Then, best individuals (population size=20) from

parent population (size=20) and offspring population (size=10) would be selected and copied to the next generation and the rest would be discarded.

4.1.1 Parents Selection Methods:

The software uses the following selection methods to choose the individuals for reproduction.

Tournament selection method:

Tournament selection selects the individuals for reproduction by contesting a tournament between the individuals in a population. The individuals winning the tournament are selected for mating or reproduction. This selection method has a parameter i.e. tournament size. The user can select the size of the tournament from 2 to the population size. In the paragraph below, this algorithm is explained using a binary tournament (tournament size=2).

Let's assume that the population size is 20, offspring population size is 10 and the algorithm has to contest a binary tournament. Since the offspring population size is 10, therefore the algorithm has to select 10 parents for mating to produce 10 offspring. The algorithm would randomly select 2 individuals from the parent population, calculate the fitness of both of these individuals and the individual with a higher fitness would be selected for mating. The individual that was not selected would not be removed from the population but rather it would stay in the parent population and could be selected in the future tournaments. Since, the offspring population size is 10, this tournament would be contested 10 times to select 10 individuals for mating.

Increasing the tournament size would increase the selection pressure of this selection method since there would be a greater chance for the best individual in the population to be selected. This would decrease the algorithm convergence time but it would decrease the efficiency. In this case, there would a higher chance that the algorithm would converge to a local minima. For example if the population size

is 20 and the tournament size is 20, then each time a tournament would be contested between all the individuals in the population every time a best individual in the population would win the tournament and would be selected. The size of tournament plays a key role in efficiency of this selecting method.

Fitness Proportionate Selection Method:

Fitness proportionate selection method selects the individuals for mating based on the concept of a roulette wheel. It assigns a chunk of roulette wheel to each individual based on its fitness. The higher the fitness of an individual, the larger the chunk it would get in a roulette wheel. Once, every individual gets a portion of the roulette wheel, the wheel is spun and the individual portion to which the marker points, that individual is selected. In this case, the higher the fitness of an each individual, the higher would be the probability of that individual to be selected. This selection method is explained briefly below.

For example there are 3 individuals in the population with the fitness of 20, 30 and 50 respectively. The algorithm calculates the fitness sum i.e. 100 in this case. The algorithm then calculates the probability of each individual.

- p(individual1) = 20/100 = 0.2
- p(individual2) = 30/100 = 0.3
- p(individual3) = 50/100 = 0.5

The algorithm then assigns portion of a roulette wheel based on their probabilities:

- portion of individual 1 = 0 - 0.2
- portion of individual 2 = 0.2 - 0.5
- portion of individual 3 = 0.5 - 1.0

The algorithm then generates a random number between 0 and 1. In this example, if the number lies between 0 and 0.2 then individual 1 is selected. If the number lies between 0.2 and 0.5, the individual 2 is selected and if the number lies between 0.5 and 1.0 then individual 3 is selected. Individual 3 having the higher probability has a better chance to be selected.

This selection method has certain limitations that could decrease its efficiency. If there are a large number of individuals and there is not much difference between their probabilities then this selection method works as a random selection as the probability of each individual to be selected would be almost the same. Similarly if one individual has a very high fitness then that individual would take a major portion of the wheel and it would have a very high probability to be selected all the time. These limitations are addressed by the rank selection method.

Rank Selection Method:

Ranks selection method addresses the problems of the fitness proportionate selection method. It sorts the population on the basis of fitness and then assigning selection probabilities to individuals according to their rank rather than according to their actual fitness values.

The best individual is ranked population size-1 and the worst individual is ranked 0. For example if the population size is 3, the best individual is ranked 2 and the worst individual is ranked 0. The algorithm then uses the following formula to select selection probabilities of individuals.

$P_{ind\text{-}rank(i)} = (2\text{-}s)/u + 2_i(s\text{-}1)/u(u\text{-}1)$ [22]

In the above formula ($1.0 < s \leq 2.0$) and (u=population size). The user can select both the values using the software in rank selection method. [22] Shows the comparison between selection probabilities for a population size of 3.

4.1.2 Recombination:

The next paragraphs briefly describes the recombination or crossover methods that the user can select to run the algorithm. All the crossover techniques generate two offspring from two parents.

One point Crossover:

One point crossover is the simplest crossover method. It was proposed in [22]. A random integer is chosen between 0 and number of genes-1.This integer is treated as a crossover point. Genes are exchanged after the crossover point. One point crossover is explained with the help of a figure in [22].

Arithmetic Crossovers:

There are three basic types of arithmetic crossover techniques that are discussed briefly below. In all of these techniques, the user enters a value of $0 < \alpha < 1$. When $\alpha=0.5$, then they are called uniform arithmetic crossovers.

Arithmetic Simple Crossover:

It generates the crossover point similar to the way it was generated in one point crossover. It then transfer all the parent genes to the offspring before the crossover point. The average of parent genes values ($\alpha=0.5$) after the crossover point is transferred to the offspring. So, the first half of each offspring is similar to the parent genes and the second half contains the average genes values of both the parents. This crossover technique is explained with the help of a figure in [22].

Arithmetic Single Crossover:

This crossover technique calculates the average ($\alpha=0.5$) of parent genes at the crossover point and transfer it to the offspring. Rest of the genes remain the same as parent genes in both the offspring. This crossover technique is explained with the help of a figure in [22].

Arithmetic Whole Crossover:

This technique works by taking the weighted sum of the two parent genes. If $\alpha=0.5$ then both the offspring produced will be identical. It uses the following formulas to calculate the gene values for the offspring.

Offspring1 = $\alpha\, x' + (1-\alpha)\, y'$

Offspring 2= $\alpha\, y' + (1-\alpha)\, x'$

[25] Explains this operation with the help of a figure.

4.1.3 Mutation:

Crossover methods described above produces new offspring. The process of mutation is performed after the recombination that modifies the gene values in the offspring. Recombination only shuffles the genes between the parents to generate the offspring. It does not introduce any new genetic material in the population. Recombination exploits all the possibilities of genes value by reshuffling them. Mutation is different. It introduces the new genetic material in the population. It explores new genetic material in the given search space. It makes sure that the population should not get stuck in the local minima.

The process of mutation involves generating a number according to a probability distribution function and then adding this number to the gene value of the offspring. The software allows the user to select between two probability distribution functions i.e. normal distribution and Cauchy distribution [13] [14].

4.1.4 Recombination and Mutation Rates:

Recombination and mutation rates describe the probabilities that these processes would take place. A higher rate would mean the higher probability for these operations to take place. User can select a value

between 0 and 1. For example if the user selects 0.8, then it would mean an 80% probability that these operations would take place.

4.1.5 Survivor Selection Method:

As discussed previously, the algorithm uses Truncation Selection as a survival selection method. It selects the best individuals among parent and offspring population and pass them on to the next generation. All other are discarded.

4.2 Particle Swarm Optimization:

Particle Swarm Optimization algorithm uses the same approach used by genetic algorithms but it evolves by a different method. It does not apply biological processes such as crossover and mutation on population to evolve and it does not select best individuals and discard the weak ones. In PSO, every particle has a certain position and velocity. In each successive iteration, the particles change their positions with respect to their velocities. The main driving force in PSO that evolves the solution is the particle's velocity. Velocity of particle is updated based on certain rules. There are different variations of PSO based on velocity update rule. Normally, velocity is updated based on particle's velocity in previous iteration, velocity and position of globally best particle (best particle in all iterations) and locally best particle (best particle in current iteration) in a swarm (population). The position of best particles in a swarm in each iteration evolves the swarm towards a better solution. Like genetic algorithms, PSO algorithm stops when a best solution is found or after a certain number of predefined iterations.

4.2.1 Velocity Update Rules:

Velocity update rule play a key role in finding the optimum solution to an optimization problem using particle swarm optimization algorithm. Inertia component (ω) introduced in [20] acts as a mechanism to control the contribution of previous velocity by adjusting inertia weight in the range 0 and 1. The

algorithm divides ω in equal steps ranging from 0 to the number of time steps. In each time step this value of ω is used to calculate the velocity of the particle.

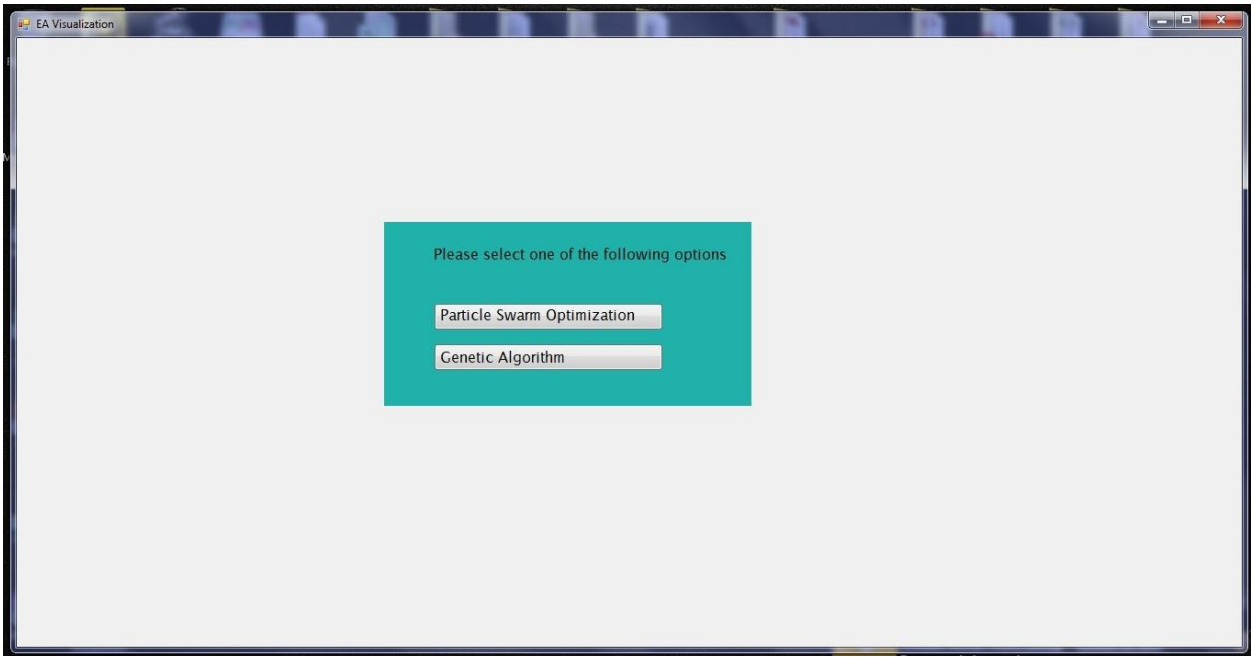
User can select between two different velocity update rules i.e. linear decreasing velocity update rule and linear increasing velocity update rule. The formula that is used to calculate the velocity of the particle is same in both these methods, however they use different values of ω in the formula in each time step. The formula that calculates the particle velocity as described in [20] is:

$$v_i(t) = \omega \cdot v_i(t-1) + c_1 \cdot r_1 \cdot (pbest(t-1) - x_i(t-1)) + c_2 \cdot r_2 \cdot (gbest(t-1) - x_i(t-1))$$

For linear decreasing, the value of ω is large in the initial time steps. It means it gives greater importance to the particle previous velocity as compared to the global best velocity. This means this method encourages exploration in the early stages of the algorithm and allow the particles to search for better solutions in the given search space. In the later iterations, the value of ω becomes smaller and smaller and thus it encourages more exploitation and allows the algorithm to converge. In case of linear increasing inertia, the value of ω is smaller in the early iterations and greater in the later iterations. It means it encourages exploitation earlier and exploration later.

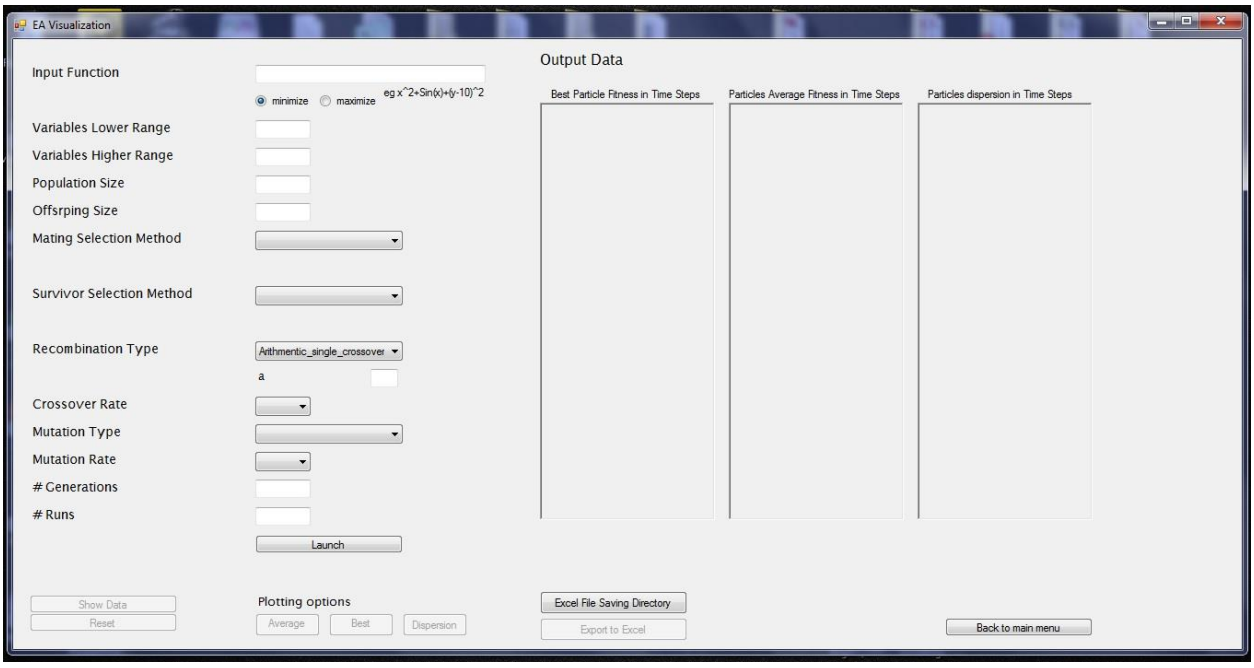
5. Graphical User Interface:

This section briefly introduces the graphical user interface of the software.



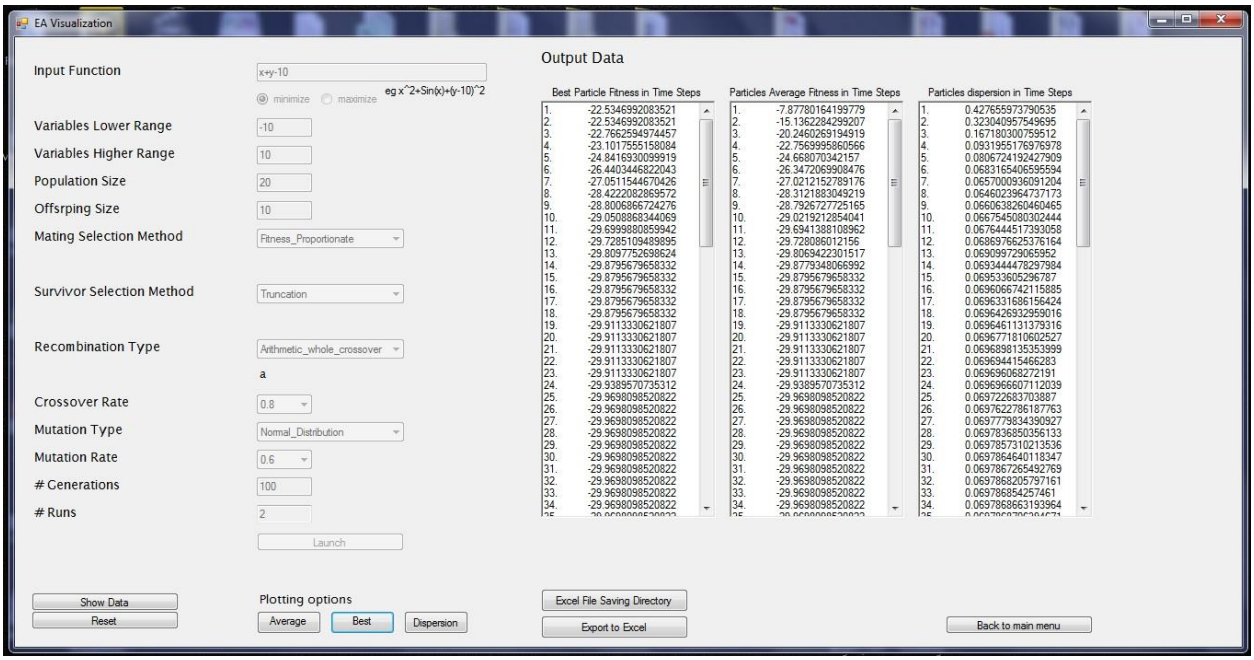
The figure above shows the main interface of the software. The user can select either particle swarm optimization or genetic algorithm to optimize a given mathematical function.

The following window is launched if the user selects genetic algorithm.

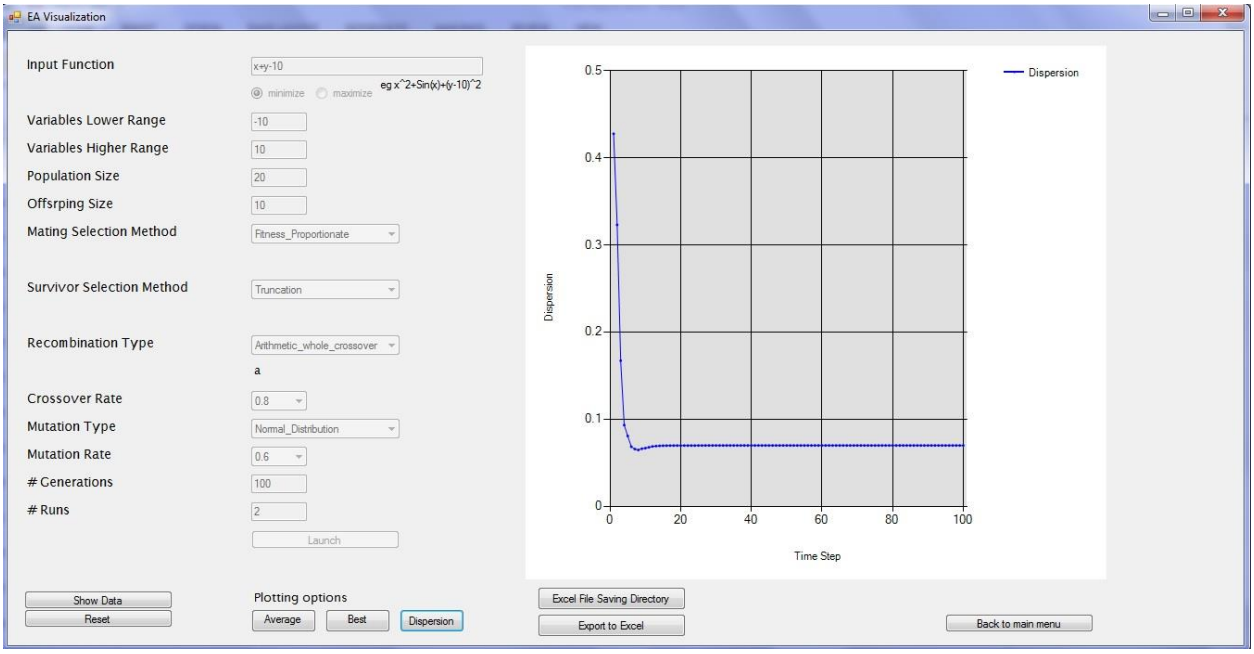


Left hand side of the interface contains all the input parameters for the genetic algorithm. The user can select variety of different values for these parameters as discussed in the previous sections. The right side of the window would show the results when the algorithm is run. Some of the buttons are inactive that would become active once the results are available to view.

The figure below shows a snapshot of the interface when the algorithm is run for the simple mathematical function i.e. x+y-10.

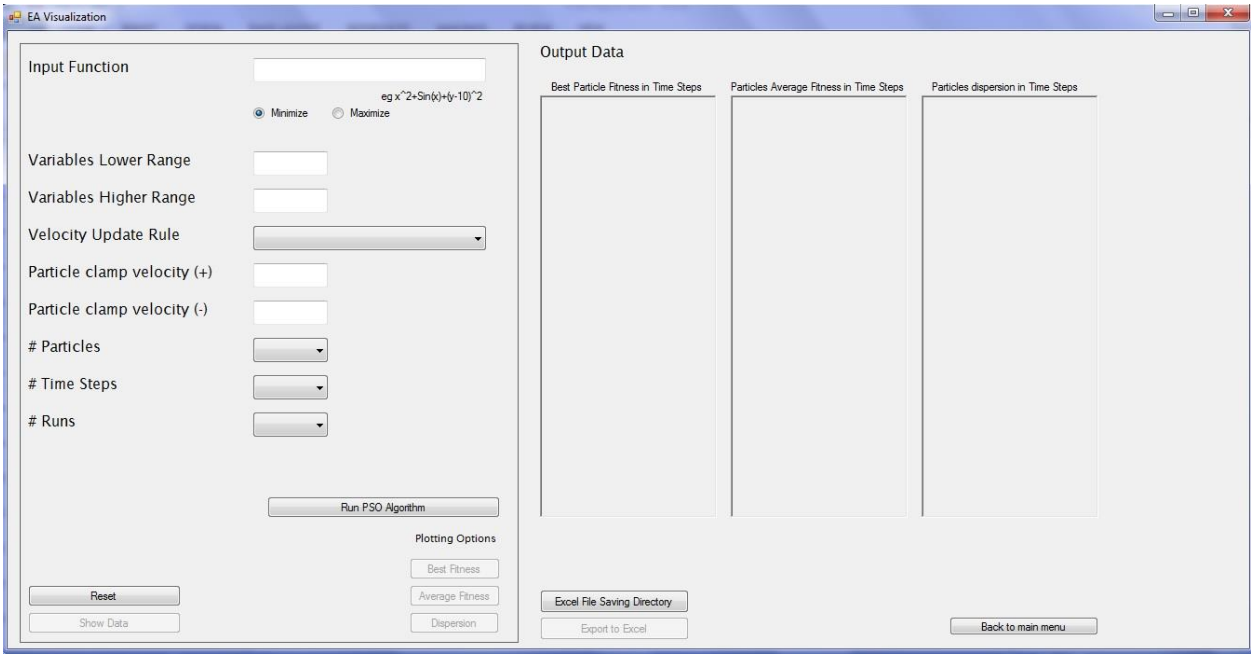


The results are shown on the right hand side of the window. The interface components that take the input parameters for the algorithm are now inactive. They can be reactivated by pressing the reset button that would reset everything. Buttons for plotting options are now active and the user can press them that would show the respective plots. The figure below shows a snapshot when the user clicks the dispersion plotting button.



The output data on the right side is now replaced with the plot of the data. The user can view the data by clicking on the 'show data' button. The user can also click on 'export to Excel' button to export all the data and plots to an excel file. The user has to click on the 'file saving directory' to select the directory where the file would be saved and then click on 'export to excel' button to generate an excel file containing the data in the selected directory.

The figure below shows the interface when the user clicks on 'Particle Swarm Optimization' button from the main interface. This interface is similar to the interface of genetic algorithm. The parameters of genetic algorithm are replaced by the parameters of the particle swarm optimization algorithm. The results are shown in the same windows as earlier. Similarly, user can export data to excel, view plots and reset the algorithm.


EA Visualization
Input Function
eg x^2+Sin(x)+(y-10)^2
Minimize
Maximize
Variables Lower Range
Variables Higher Range
Velocity Update Rule
Particle clamp velocity (+)
Particle clamp velocity (-)
Particles
Time Steps
Runs
Run PSO Algorithm
Plotting Options
Best Fitness
Average Fitness
Dispersion
Reset
Show Data
Output Data
Best Particle Fitness in Time Steps
Particles Average Fitness in Time Steps
Particles dispersion in Time Steps
Excel File Saving Directory
Export to Excel
Back to main menu

Conclusion:

The report discussed different evolutionary method and their parameters that are used for different optimization problems. The report provided a detailed description of a software that could be used to optimize any mathematical function using genetic algorithm and particle swarm optimization algorithm. This report provided a detailed description of all the algorithms that are used by the software. It discussed in detail different parameters that the user can select to run the algorithm. The report also introduces the graphical user interface of the software to help the users to make themselves familiarize with the user friendly graphical user interface of the software.

References:

[1] Zitler, E., & Thiele, L. (1999). *Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach*. IEEE Transactions on Evolutionary Computation, 3(4).

[2] Eiben, A.E. (1999). *Parameter control in evolutionary algorithms*. Evolutionary Computation, IEEE Transactions, 3(2), 124-141. doi:10.1109/4235.771166

[3] Vesterstrom, J., & Thomsen, R. (2004). *A Comparative Study of Differential Evolution, Particle Swarm Optimization, and Evolutionary Algorithms on Numerical Benchmark Problems*. IEE Press.

[4] Budin, L., Golub, M., & Budin, A. *Traditional Techniques of Genetic Algorithms Applied to Floating-Point Chromosome Representations.*

[5] Chudasama, C., S. M. Shah, & Panchal, M. (2011). *Comparison of Parents Selection Methods of Genetic Algorithm for TSP*. Proceedings Published by International Journal of Computer Applications® (IJCA) International Conference on Computer Communication and Networks CSI- COMNET-2011.

[6] Alabsi, F., & Naoum, R. (2012). *Comparison of Selection Methods and Crossover Operations using Steady State Genetic Based Intrusion Detection System*. Journal of Emerging Trends in Computing and Information Sciences, 3(7).

[7] Razali, N., & Geraghty, J. (2011). *Genetic Algorithm Performance with Different Selection Strategies in Solving TSP*. Proceedings of the World Congress on Engineering, II.

[8] Kumar, R., & Jyotishree. (2012). *Blending Roulette Wheel Selection & Rank Selection in Genetic Algorithms*. International Journal of Machine Learning and Computing, 2(4).

[9] Magalhaes-Mandes, J. (2013*). A Comparative Study of Crossover Operators for Genetic Algorithms to Solve the Job Shop Scheduling Problem*. IEEE Press, 12(4).

[10] Vekaria, K., & Clack, C. *Selective Crossover in Genetic Algorithms: An Empirical Study*.

[11] Kaya, Y., Uyar, M., & Tekin, R. *A Novel Crossover Operator for Genetic Algorithms: Ring Crossover*.

[12] Chiung, A., Jongsoo ,A ., Gyunghyun, A ., & Yoonho, B. *An efficient genetic algorithm for the traveling salesman problem with precedence constraints*.

[13] Normal Distribution. (2001). Retrieved from http://www.encyclopedia.com

[14] N. L. Johnson, S. Kotz, and N. Balakrishnan (1994). *Continuous Univariate Distributions*. New York: Wiley, 1(16).

[15] Abdoun, O., Abouchabaka, J., & Tajani, C. *Analyzing the Performance of Mutation Operators to Solve the Travelling Salesman Problem.*

[16] Libelli, M., & Alba, P. (2000). *Adaptive mutation in genetic algorithms*. Soft Computing, 4, 76-80.

[17] Clerk, M. (2005). Particle Swarm Optimization.

[18] James Kennedy and Russell Eberhart. *Particle swarm optimization*. In Proceedings of IEEE International Conference on Neural Networks, pages 1942–1948, Piscataway, NJ, USA, 1995. IEEE Press.

[19] Bonyadi, M., Michalewicz, Z., & Li, X. (2013). *An analysis of the velocity updating rule of the particle swarm optimization algorithm*. doi:10.1007/s10732-014-9245-2

[20] Shi, Y., & Eberhart, R. (1998). A Modified Particle Swarm Optimizer. IEEE Press.

[21] Venter, G., & Sobieski, J. (2003). Particle Swarm Optimization. AIAA Journal, 41(8), 1583-1589.

[22] Eiben, & Smith. (2003). *Genetic Algorithms*. In Introduction to Evolutionary Computing. Page 47-61