December 2018

Conversion of a Simulator Written in C++ to JS and Optimizing the Simulation Parameters Using Evolutionary Algorithms

Venkata Sivasai Pavan Kumar Lottala
University of Wisconsin-Milwaukee

CONVERSION OF A SIMULATOR WRITTEN IN C++ TO JS AND OPTIMIZING THE SIMULATION PARAMETERS USING EVOLUTIONARY ALGORITHMS

by

Sai Pavan Kumar Lottala Venkata Siva

A Thesis Submitted in
Partial Fulfillment of the
Requirements for the Degree of

Master of Science
in Computer Science

at

The University of Wisconsin-Milwaukee
December 2018

ABSTRACT

CONVERSION OF A SIMULATOR WRITTEN IN C++ TO JS AND OPTIMIZING THE SIMULATION PARAMETERS USING EVOLUTIONARY ALGORITHMS

by

Sai Pavan Kumar Lottala Venkata Siva

The University of Wisconsin-Milwaukee, 2018
Under the Supervision of Professor Dr. Mukul Goyal

A discrete event simulator written in C++ is converted in to Java Script, that tracks the blood glucose level of a person in response to a timed sequence of diet and exercise activities. Its main objective is to capture the average impact of the various diet/exercise activities on the blood glucose level. The main aim for translation of the code in to Java Script is that the simulator can be hosted on the Google Firebase Cloud and can be available for the public use. The optimization of the simulator parameters such as excretionKidney_impact, glut4Impact_, glycolysisMinImpact_, gngImpact_, peakinsulinLevel_, glycolysisMaxImpact_, liverGlycogenBreakdownImpact_ and liverGlycogensyntheis_Impact is done using evolutionary algorithms, where the simulator is given base blood glucose level and peak blood glucose level as the input parameters to the simulator. The output produced from the evolutionary algorithms are compared and the best one is recommended.

Dedicated to my parents

for their love, motivation and support

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ES	ECMAScript
BGL	Blood Glucose Level
JS	Java Script
LLVM	Low Level Virtual Machine
PSO	Particle Swarm Optimization
BSO	Backtracking search Optimization

1. INTRODUCTION

According to the latest report by the Centers for Disease Control and Prevention, more than 100 million U.S adults are now living with diabetes or prediabetes. As per the report in 2015, around 9.4 percent of the U.S. population has diabetes. Another 84.1 million have prediabetes, which if not treated will lead to type 2 diabetes within five years [12]. People with type-2 diabetes have a minimum capacity to produce insulin, but their bodies develop insulin resistance and hence are not able to react strongly to keep their blood glucose level under control, even when the insulin is present in their blood. The people with Type-1 diabetes must receive insulin by external means since they cannot produce the insulin endogenously at all. The presence of a high level of BGL in blood for a long time will result in heart/kidney failure, blindness and limb amputations. People with diabetes should plan their food and exercise carefully so that they can keep there BGL under control and lead a happy life. This simulator aims towards helping people to plan their activities carefully and monitor their BGL minute by minute so that they can keep the BGL under control. This simulator is based on the discrete event model where the time increments, in units called ticks, are one minute long and at the beginning of each tick, the simulator will use the food/exercise events that are present and directs the organs to do work similar to the organs in the human body during this tick. All the food/exercise events are given by the user to the simulator.

The entire Thesis is divided in to two parts. The first part deals with the translation of a simulator written in C++ in to Java Script, process followed in the conversion of the code, problems faced during the conversions were discussed in detail. The type of Java Script used is ES 6 which gives us more features likes array functions, classes, methods which are more readable and like objected oriented programming. The main problems faced during the translation are language problems, updating problems, debugging problems and availability of the packages for the simulator.

The next part is to find the best values for the simulation parameters such as excretionKidney_impact, glut4Impact_, glycolysisMinImpact_, gngImpact_, peakinsulinLevel_, glycolysisMaxImpact_, liverGlycogenBreakdownImpact_ and liverGlycogensyntheis_Impact so that when a user gives his target base blood glucose level and peak blood glucose level the algorithms find the optimum values for the above parameters in minimum time to get the output within the stipulated range. The normal brute force methods are not useful and so in order to achieve our targets we need to make sure to limit our search space and reach our target in less time. The general strategy is that we give the input values randomly and check the output with an optimization strategy to help search for the optimal solution. This will guide the change in the input parameters in to the simulation model so that we can reach our targets in less time. The normal optimization methods that are performed are Gradient based search methods, stochastic optimization, response surface methodology, Heuristic Methods, A-teams and statistical methods. The best results for the simulator can be achieved by the Heuristic Methods, which are the latest developments and best suited for the simulator.

The simulator uses Evolutionary algorithms which imitate the principal of natural evolution as a method to solve the parameter optimization problems. Two optimization algorithms were used in order to achieve the best results. Particle swarm optimization and Back tracking search optimization, which are some of the best optimization algorithms and are simple, take less time to achieve the result and perfectly satisfy our constraints. Towards the end both the algorithms were compared and the best one is recommended to optimize the parameters in the simulator.

2. Overview of the Process followed in the conversion of Code from C++ in to JS

The entire process of conversion of the code is divided in to three parts. The first part is the conversion of all the body organs such as Blood, Kidneys, Adipose Tissue, Brain, Heart, Portal Vein, Stomach, Intestine, Liver and Muscles were done except Human body. Once all the organs were translated then the Human body was translated since it is the gateway from which we call all the organs, specific methods for adding, reading and processing the food, exercise events were implemented in the human body which were not present in the original code because of the elimination of the simctl object. All the methods that are present in the simctl object in the original code were implemented in the human body object.

The overview of the human body object which is the most important part of the simulator is as follows. The human body object contains three parts. The first part takes the input data in the form of food event, exercise even and process them in to the priority queues. The second part of the human body maintains the time and fires the events in the priority queue in the order of their firing times. The third part maintains the other objects such as Intestine, blood, stomach, portal vein, liver, kidney, muscles, adipose tissue, brain, heart which are activated at the beginning of each simulation. At the beginning of each simulation after the first and second parts does their work, this object reads the different values from the food and exercise events, including the different parameters that affect the different objects and calls the other objects. The third part also contains methods that cause the food to be added to the stomach and update the energy needs, when the first part fires an exercise event. Human body has the cognitive ability to see if the stomach has some undigested food or not, If the body is doing some exercise or not. There are

four variables to determine the above states: Fed Resting, Fed Exercising, Post Absorptive Resting and Post Absorptive Exercising. These four variables allow the configurable parameters to take different values which instate help in controlling the other organs. The priority queue object was changed when compare to its original code so that it can be used in hosting the project on google firebase.

The third part is the use of packages which are needed for the simulator. Since there are only limited number of packages available in java script some of the packages are implemented manually. In other cases, like pseudo random generator even if we implement the code in java script it is not efficient so C++ addons were used when the packages required were not available in Java script. All the parts were combined in to a single file so that it can be easily hosted on the firebase and eliminates the usage of import statements in all the files. All the above were implemented using the ES6. ES6 was used since it has simple syntax, more readable, and more features were added like arrow functions, string functions, map objects, classes etc., It is like objected oriented language syntax which makes it very easy to read, understand and debug.

2.1 IMPLEMENTATION OF THE ORGANS IN JAVA SCRIPT

All the organs such as Blood, Kidneys, Adipose Tissue, Brain, Heart, Portal Vein, Stomach, Intestine, Liver, Muscles and Human body were implemented using the standard ES6 class implementation. Each class consists of three parts. The first part contains constructor, here all the values for a class are initialized. The second part contains process Tick method from which all the remaining methods in the class are called. The third part contains additional methods in the class which are not part of the first and second parts which can be called from inside and from outside of the class, provided the class is initialized. In the third part setParams method can be taken as the best example which can be used in changing the default parameters of the class.

2.2 INPUT FOOD AND EXERCISE PARAMETERS DESCRIPTION

For this simulator, the input food parameters are described in terms of item number, name of the food, serving size, amount of rapidly available glucose, slowly available glucose, protein and fat per serving. The addFoodType method in the human body is used for inputting the parameters in to the simulator. The rapidly available glucose contains sugars and rapidly digestible starch. The slowly available glucose contains slowly digestible starch. The exercise parameters are given in terms of exercise number, name of the exercise and its intensity in units, of METs with 1 MET is 1 kcal of energy expenditure per kg of body weight per hour. The addExerciseType method in Human Body is used for inputting the Exercise parameters in to the human body.

2.3 TYPES OF PROBLEMS ENCOUNTERED

There were mainly four types of problems experienced while converting the code from C++ in to Java Script. They are language problems, updating problems, debugging problems and availability of packages. The language problems stem from the simulator are from the use of biological names which sometimes leads to confusion for example glut4Impact and glutImpact where only number four is missing in the second variable. The presence of many comments in the code, use of underscores for some variables also contributed to the confusion in the code. There were some problems from the naming of the variables in the original code because the use of same names in the Java Script is not allowed and might lead to the crashing of the code. While the code is being translated in to Java Script, updating the original code has led to some of the major problems.

The updating of the translated code has become particularly hard since there was no documentation on the changes made in the original code. So, for each updating entire code has to be compared with the complete C++ code to get the Java Script code up to date. In some cases,

more methods were added in the code which uses some standard packages in C++ which are not available in Java script. There was no effective debugger available to debug the entire code when the code is being translated part by part. The only method available is the use of print statements to see if the code translated is correct or not. It was harder to debug since some of the packages which were not available in Java Script were compensated with the use of the C++ Addons which were practically almost impossible to debug. Since the code runs for a long amount of time to get the result and prints a ton of statements it is also impossible to find the error if the error occurs after hundred iterations. The built-in debuggers in browsers also are not useful because of the above reasons.

The availability of the packages for the Java Script has become the major problem in the conversion of the code. Some of the libraries in C++ like math library, stdlib, apache library are not available in Java Script. In order to compensate for the missing libraries some of the libraries were manually implemented. In some cases, like the pseudo random generator is the biggest problem since the implementation similar to C++ will be more time consuming and not efficient at all. More than 20 to 25 npm packages were used to get the similar result produced by the random generator in C++ but to no avail. The similar problem occurred for the use of poisson distribution from the apache library.

In order to overcome the above problems C++ addons were used. The C++ addons have higher performance, can have access to all the C++ libraries. nbind package is used in calling the C++ files that contains the pseudo random generator, then the files are compiled to asm.js which in turn can be run on the browsers or node JS server. The files are compiled to asm.js using emscripten, it is built using the LLVM, that lets user run C and C++ on the web at a good speed without any plugins.

2.4 RECOMMENDATIONS FOR AN EFFECTIVE CONVERSION

In order to overcome the above problems, the following steps are recommended so that it can be easy for a code to be converted from one platform to another. The language problems can be removed by using simple names and by using of good symbols which are less confusing. The comments in the middle can be eliminated instead of that a good documentation will provide more help for solving the language problems and when there is an update, we can simply give information in the document which will greatly reduce the time for updating the new code. A separate documentation for the packages will help in finding the packages that are not available in cross platform library, which can then be obtained either by writing the complete package in the new platform or finding a work around way like C++ addons. If the dependency on the packages and libraries is decreased, then the debuggers will help in finding out the problem quickly. Instead of depending on traditional debuggers like browsers it is useful to use cross platform debuggers like visual studio, brackets etc.,

If the packages or libraries are not available in the other platform it is better to see if it is efficient to implement them in the missing platform since if it is not efficient there might be other options like addons that are available which will help in solving the problem, only after trying all the other possibilities it is better to implement if there is no other option available. It is always better to use simple data structures that are easy to implement and that are efficient than those that are complex and not efficient. If the above recommendations are followed it is easy to convert a code from one programming language to another programming language efficiently in a small amount of time.

3.OVERVIEW OF THE OPTIMIZATION PROCESS

Optimization is the process by which one finds the maximum or minimum value of a function. Maximization of a function f is similar to minimization of the opposite of this function, $-f$ [11] .In mathematically a minimization task and maximization task is referred as follows [10, 11]:

Minimization Task:

$$Given\ f : R^n \rightarrow R$$

$$\text{Find}\ \ \hat{x} \in R^n\ such\ that\ f(\hat{x}) \leq f(x), \forall\ x \in R^n$$

Maximization Task:

$$Given\ f : R^n \rightarrow R$$

$$\text{Find}\ \ \hat{x} \in R^n\ such\ that\ f(\hat{x}) \geq f(x), \forall\ x \in R^n$$

From the above the domain of f is R^n which can be called as parameter space or search space. there can be many solutions to the function f but $\hat{x}$ is the best optimal solution in the search space R^n. The value n refers to number of dimensions of the search space and thus the number of parameters involved in the optimization problem. The function f from the above is called the objective function which takes the input parameters and gives out the result, which is usually a one dimension.

The optimality for the set of the parameters depends on this fitness value. For a differentiable function f, maxima and minima can be easily found out but since the simulator is a black box it is not possible to find the maximum and minimum values in the normal way. It is to this black box that we apply the input parameters and the result we get from the black box is value that needs to be optimized.

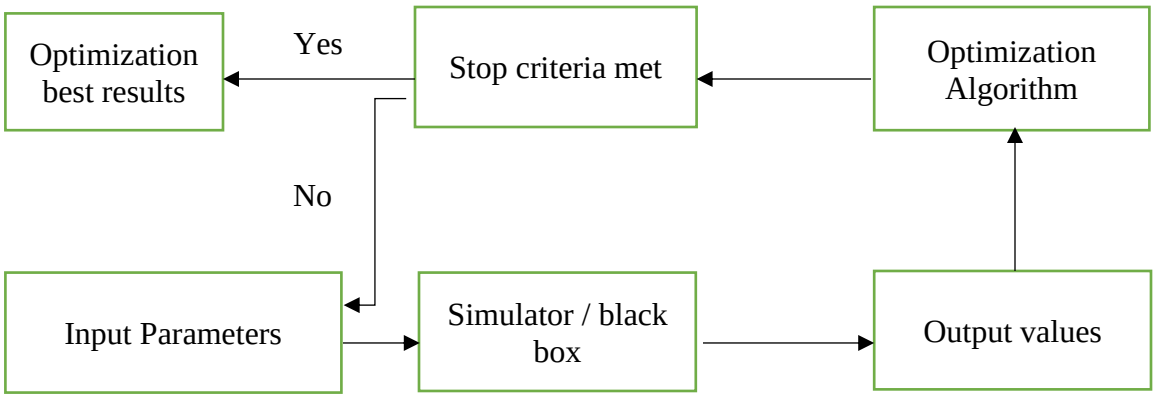

Figure 1. Optimization Process

3.1 OVERVIEW OF PARTICLE SWARM OPTIMIZATION

Particle swarm optimization is one of the best optimizations and simple evolutionary algorithm that helps in exploring the search space of a given problem to find the parameters that achieve the global maximum or global minimum in an optimal amount of time. The main idea for the algorithm is the idea of swarm intelligence based on the observation of swarming objects by certain kinds of animals and the field of evolutionary computation.

Initially PSO algorithm randomly chooses the candidate solutions (best parameters) in the search space, the number of candidate solutions depends on the user. During each iteration of the algorithm the candidate solution is evaluated by the objective function being optimized, determining the fitness of that solution. Each candidate solution can be taught as a particle flying through the fitness landscape (curve generated by the objective function) finding the maximum and minimum of the objective function. It should be taken in to consideration that PSO has no information about the underlying curve generated by the objective function hence there is no way of predicting whether any of the candidate solutions are near to or far away from the local or global maximum/minimum.

The PSO algorithm simply uses the objective function to evaluate the candidate solutions and operates up on the resultant fitness values. Each particle of PSO has three main parts that it maintains. They are the particle position, fitness value and particle velocity. Particles also remembers the best fitness value it has achieved so far which is referred to as the individual best fitness value and the corresponding position is referred to as the individual best position so far. PSO algorithm also maintains the global best fitness value achieved so far from all the particles so far and the corresponding global best position.

The algorithm performs three main steps until the stop conditions are met. The three main steps are as follows [11]:

A. Calculate the fitness value (obtained from the objective function) of each particle
B. Change the individual, global best fitness values and positions after each iteration
C. Update the velocity and the position of each particle based on the above calculations.

From the above the first two steps are as follows: fitness evaluation is obtained by the giving the candidate solution as the input parameters to the objective function. Individual and global best fitness values and positions are obtained by comparing the newly found fitness values against the previous individual and global best fitness values and replacing the best fitness and positions as necessary.

The velocity and the position update of each particle is done in the following ways. The velocity of each particle in the swarm is updated using the following equation:

$$v_i(t+1) = wv_i(t) + c_1 r_1[\hat{x}_i(t) - x_i(t)] + c_2 r_2[g(t) - x_i(t)]$$

The above equation is used by each particle with index i representing the index of each particle. The $w\ (0.8 \leq w \leq 1.2\ ), c_1\ (c_1 \simeq 2), c_2\ (c_2 \simeq 2)$ are user defined constants and $r_1, r_2\ (0 \leq r_1 \leq 1\ and\ 0 \leq r_2 \leq 1\ )$ are random values regenerated for each velocity update.

$v_i(t)$: velocity of particle i at time t

$x_i(t)$: position of particle i at time t

$wv_i(t)$: It is called the inertia component which is responsible for making the particle move in the same original direction. The coefficient w is responsible for either dampening or accelerating the particle movement in its original direction. The lower values helps in getting the result quickly and the higher values are better for exploring the search space.

$c_1 r_1[\hat{x}_i(t) - x_i(t)]$: It is called the cognitive component. It is used as the particle's memory, causing it to return to the regions of search space in which it has experienced high individual fitness and generally affects the particle's step size towards its best individual position or candidate solution.

$c_2 r_2[g(t) - x_i(t)]$: It is called the social component makes the particle to move to the best region of the search space found so far.

In order to make sure that the particles do not move beyond the boundaries or constraints of the search space, velocity clamping needs to be done by limiting the maximum velocity of each particle. For search space bounded by the range $[-x_{max}, x_{max}]$, the velocity clamping limits the velocity to the range $[-v_{max}, v_{max}]$, $v_{max} = k \times x_{max}$. The value of the k can be in the range of 0.1 and 1.0 which is generally user defined.

The particle position can be updated as follows:

$$x_i(t+1) = x_i(t) + v_i(t+1)$$

The above steps are repeated until the stopping conditions are met as shown in the figure below. The general stopping conditions include limiting the number of iterations, difference in the result obtained from the candidate solutions is within the preferred range. The algorithm is written in

python and the result is calculated in this simulator by combining both the stopping conditions ways that is described above.

3.2 IMPLEMENTATION OF PARTICLE SWARM OPTIMIZATION

The entire particle swarm optimization is implemented by python. Only NumPy package was used. All the initial constants that are given at the start of the program are number of particles, omega, c1, c2, max_iterations, minstep, minfunc, lowerbound, upperbound. All the above discussed values are given default but can be changed in order to suit the user. For each particle, the velocity clamping is given at the start of the program and each particle velocity, position and the best position and the corresponding function values are stored in the separate NumPy arrays. Then the particles are moved along the curve of the objective function with velocity (calculated as discussed in the modelling section) to get to the new positions along the curve with the best position and best fitness value achieved so far is stored by the particle. Once the iterations are done the best position and the corresponding best fitness value achieved by each particle are returned at the end from which the target value and the corresponding position values are selected.

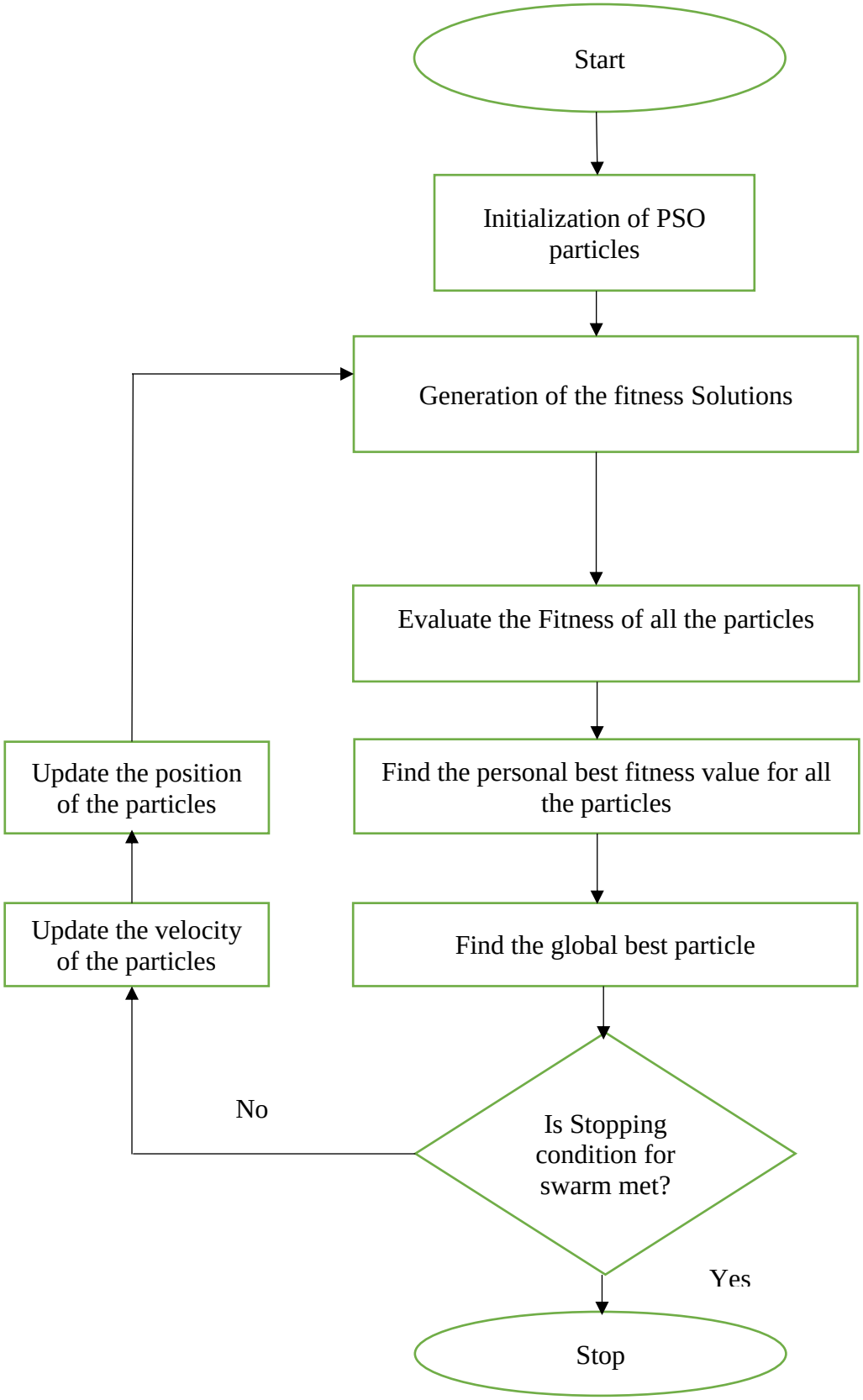

Figure 2. Flow chart for Particle Swarm Optimization

3.3 BACKTRACKING SEARCH OPTIMIZATION ALGORITHM

Backtracking search optimization algorithm is an iterative population based evolutionary algorithm used to find the global minimum. BSA can be divided in to five steps: Initialization, selection-I, mutation, crossover, selection-II [2, 3, 4]. There are two types of population in the BSA: evolution population and trial population. The historical information regarding the evolution population is composed in the trial population. There is a search direction matrix is built by the trial population and the evolution population to update the positions of individuals. The general structure of the BSA is as follows:

A. Initialization: BSA initializes the initial population *P* and the historical population $oldP$ as follows:

$$P_{ij} \sim U(low_j, up_j)$$

$$oldP_{ij} \sim U(low_j, up_j)$$

Where *i* = 1,2, 3,…,N (N is the population Size) and *j* = 1,2,3,4,…….,D(D is the problem dimension or parameter dimension), U is the uniform distribution and each P_i is the target individual in the population P.

B. Selection-I: In this stage the $oldP$ is introduced in BSA by the following conditions:

$$if\ a < b\ then\ oldP := P|a, b \sim U(0,1)$$

Where: = is the update operation that the population belonging to a randomly selected previous generation as the historical population to be used in the generation of a search direction matrix, which allows taking advantages of old experiences to generate a trial population and remembers the historical population until it is changed, which resulting BSA to have memory. Once $oldP$ is calculated then the order of the values in $oldP$ is randomly changed.

$$oldP := permuting(oldP)$$

C. Mutation: The initial trail population mutant is calculated using the following equation:

$$Mutant = P + F \times (oldP - P)$$

F controls the amplitude of the search amplitude with value being set by user. Since BSA uses the historical population to calculate the search direction, BSA generates a trail population by taking advantage from the previous generations. The value of F can be found out by trying repeated values and selecting the best value from the options that have been tried.

D. Crossover: In this stage the final form of the trial population T is generated. Mutant from the mutation process is the initial value for the cross over. Two strategies are used in the crossover to define the BSA's map. A binary integer-valued matrix (map) of size N X D is calculated which indicates the individuals of the trial population T that needs to be manipulated by using the relevant individuals of current population P, the individuals of T are updated only when the following condition is satisfied:

$$if\ map_{n,m} = 1\ where\ n \in \{1,2,3,4, \ldots\ldots, N\} and\ m \in \{1,2,3,4, \ldots\ldots D\}, T_{n,m} \coloneqq P_{n,m}$$

the first strategy is the use of mix rate parameter that controls the number of elements of individuals that will mutate in a trial, the other strategy uses only one randomly chosen individual to mutate in each trial. In order to make sure that the trial population values obtained at the end of the crossover process is within the search space checks are performed at the end to see if the trial populations generated by the crossover is within the limits if not then those trial population values are regenerated.

E. Selection-II: In BSA's second selection process, if the fitness value of trail population individual T_i is better than that of original population P_i then $T_i$will update P_i , once the above process is repeated for all the current population, we select the P_{best} . Towards the end final check

is done P_{best} is compared with the current global minimum value if P_{best} is better than the global minimum value then this value is returned with the corresponding P_{best} fitness value.

The flow chart for the backtracking optimization is as follows:

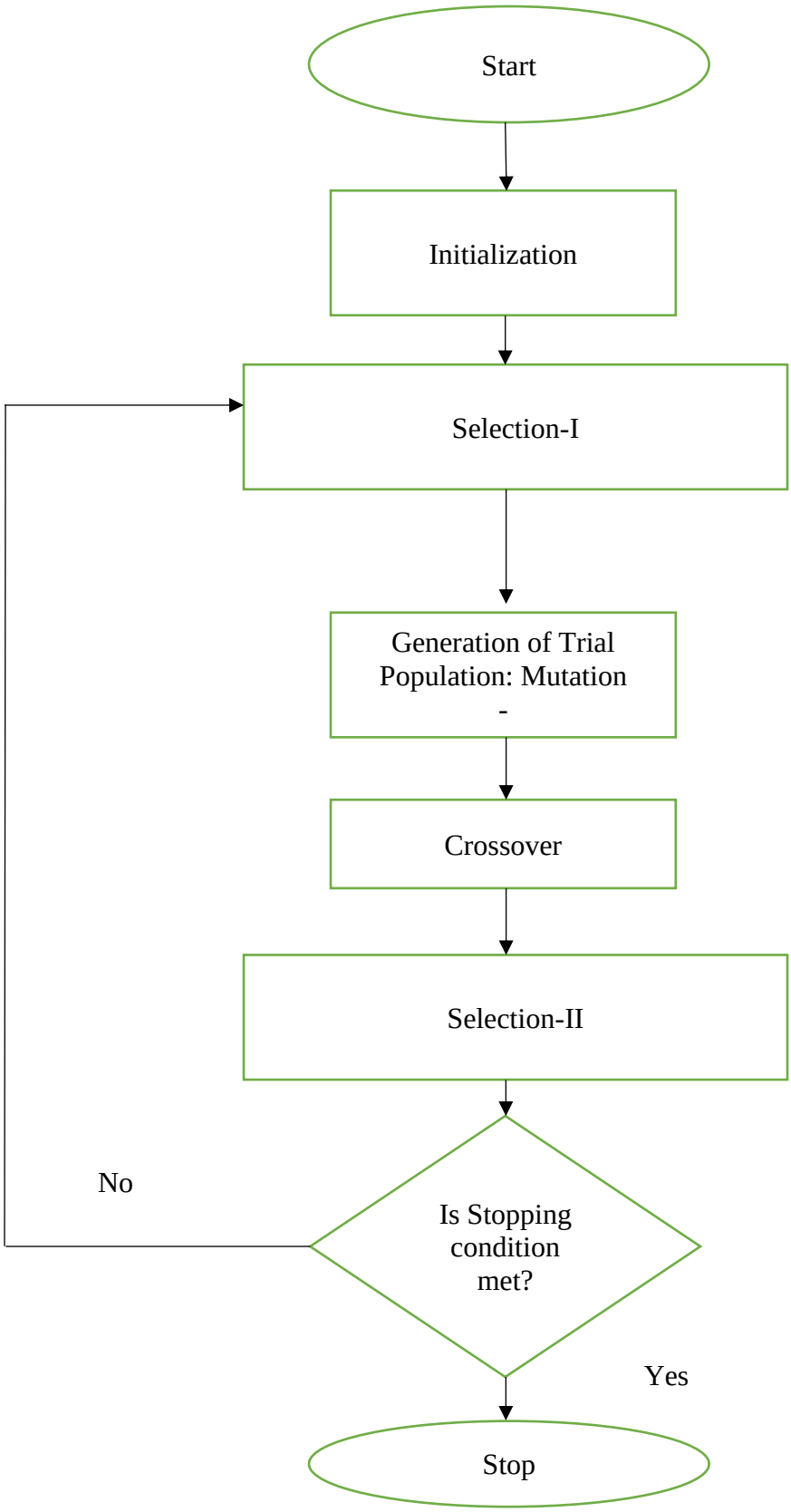

Figure 3. Flow chart for Backtracking Search Optimization

3.4 IMPLEMENTATION OF BACKTRACKING SEARCH OPTIMIZATION

The backtracking search optimization algorithm uses the following packages: NumPy, random, math. The initial parameters given by the user are the objective function, population, lower bound, upper bound, max iterations and the mixrate. All the above parameters are default but can be changed by the user. At the start the size and the dimension variable are initialized with the population and length of the upper bound. The remaining variables are initialized with the zero values with the corresponding length depending on the type of the variable. In the first part the Initial_parameter_values variable and the old_parameter_values variable is filled with the random values within the constraints as explained in the previous section. The fitness_p variable has the values obtained by calling the objective function with the initial_parameter_values and the param_values as the input variables. The selection-I stage starts with the initialization of four variables a, b, c, d with random values and then the old_parameter values are changed with the initial_parameter_values when value of a is than b. Then the permutation of the old_parameter_values take place and the mutant variable is calculated as discussed in the previous section. For the crossover stage map variable is initialized with the NumPy array with size and the dimension as its input parameters.

In the crossover stage, generation of trail population is performed as discussed in the previous section. The boundary control mechanism is implemented as follows: every value in the trail population is checked to see of it is lower than the lower bound or higher than the higher bound if it is, new values are produced to replace the current values in the trail population. Finally, selection-II stage takes place in which the fitness_t is initialized with the target values which are obtained by calling the objective function with trail population as the input parameters. Then every fitness_t value is checked with fitness_p value to see if it is less than the later, if so then that

particular value is copied in to fitness_t value and the corresponding Trail population value is copied in to the Initial_parameter_values variable. Then the best minimum and the corresponding parameters values are returned.

3.5 OBJECTIVE FUNCTION

The packages used in the objective function are subprocess, OS. The objective function is the one that calls the simulator with input parameters from the evolutionary algorithms and gives out the target values. The main methods in the objective function are run_simulator, read_file and modify_params. The run_simulator takes names of the parameters and the corresponding values as the input parameters. In the run_simulator the modify_params method is called with each parameter name, corresponding value and the file where the modification of the value needs to take place. The above process is done until all the values in the input parameters file are changed. Once the modifications of the input parameters file is complete the diabetic simulator is called with food, exercise, input parameters and events text files as the input. The output text file produced by the simulator contains the target values.

If the text file is present, then there is a method called read_file reads the file from the directory which checks for the target values in the file and returns them. The target values are converted in to float data types so that they can be used by the evolutionary algorithms. If the text file is not present, then input values to the run_simulator is changed by incrementing each value by 0.01 and the run_simulator method is called with the new input values; the above process is repeated until the simulator produces a valid text file.

4. RESULTS

The optimization algorithms used in the previous section were used in order to achieve the best optimized values to the simulation parameters. Different default values are taken and tried on different target values to test which algorithm will give the best result. The below Table 1 describes the default parameters taken for PSO algorithm.

DEFAULT PARAMETERS	DEFAULT VALUES
Number of Particles	20
omega	0.8
C1	0.8
C2	0.8
Max_iterations	30
Objective function	run_Simulator
Lower bound	[0,0,0,0,1,1,1,0]
Upper bound	[1,1,1,1,2.6,2,3,2]

Table 1: Default Parameters For PSO

The above parameters from Table 1 can be changed as per the user requirement but the values above are selected after extensive testing to see which will work best in small amount of time and gives out the best result possible. The other default variables like min step, min function is left to the user discretion. The below Table 2 and Table 3 represents the best values for the peak insulin level parameter when trying to optimize the parameter and the corresponding base blood glucose level obtained. The default values from Table 1 were used. four base BGL values were

tested with two normal person readings with target base BGL values as 85, 65 and two for diabetic person readings with target base BGL values as 210, 220

Peak Insulin Level For Person 1	Base BGL	Peak Insulin Level For Person 2	Base BGL
1.0	90.097	1.0	70.256
1.0	90.097	1.0	70.256
1.0	90.097	1.0	70.256
1.0	90.097	1.0	70.256
1.0	90.097	1.0	70.256
1.0	90.097	1.0	70.256
Total Time Taken in Seconds	76.88	Total Time Taken in Seconds	84.019

Table 2: PSO Results for Peak Insulin Level VS Base BGL for Normal Persons

Peak Insulin Level For Person 3	Base BGL	Peak Insulin Level For Person 4	Base BGL
1.0	216.315	0.45131204	221.153
1.0	216.315	0.2841238	221.153
1.0	216.315	0.00515574	221.153
1.0	216.315	0.09473472	221.153
1.0	216.315	0.39059779	221.153
1.0	216.315	0.29715696	221.153
Total Time Taken in Seconds	82.281	Total Time Taken in Seconds	47.500

Table 3: PSO Results for Peak Insulin Level VS Base BGL for Diabetic Persons

The following Table 4, Table 5 gives the optimized results for the parameter glut4Impact and the corresponding result obtained for the target value Base BGL for two normal persons and two diabetic persons

Glut4Impact_ For Person 1	Base BGL	Glut4Impact_ For Person 2	Base BGL
1.0	90.097	0.94376497	70.147
1.0	90.097	1.0	70.256
1.0	90.097	s0.93539366	70.15
1.0	90.097	0.93647323	70.15
1.0	90.097	0.91961868	70.156
1.0	90.097	0.91264966	70.158
Total Time Taken in Seconds	77.21	Total Time Taken in Seconds	76.820

Table 4: PSO Results for Glut4Impact VS Base BGL for Normal Persons

Glut4Impact_ For Person 3	Base BGL	Glut4Impact_ For Person 4	Base BGL
1.0	216.315	0.43352566	70.147
1.0	216.315	0.13864778	70.256
1.0	216.315	0.11110498	70.15
1.0	216.315	0.29124837	70.15
1.0	216.315	0.470699	70.156
1.0	216.315	0.56624767	70.158
Total Time Taken in Seconds	83.99	Total Time Taken in Seconds	48.785

Table 5: PSO Results for Glut4Impact VS Base BGL for Diabetic Persons

The following Table 6, Table 7 gives the optimized results for the parameter ExcretionKidneysImpact and the corresponding result obtained for the target value Base BGL for two normal persons and two diabetic persons

Excretion_Kidneys_Impact For Person 1	Base BGL	Excretion_Kidneys_Impact For Person 2	Base BGL
0.29096885	90.097	0.21918861	70.256
0.55928564	90.097	0.07241679	70.256
0.73581576	90.097	0.07326758	70.256
0.61736362	90.097	0.0286695	70.256
0.23467464	90.097	0.09186598	70.256
0.72257251	90.097	0.97044313	70.256
Total Time Taken in Seconds	75.279	Total Time Taken in Seconds	81.251

Table 6: PSO Results for Excretion_Kidneys_Impact VS Base BGL for Normal Persons

Excretion_Kidneys_Impact For Person 3	Base BGL	Excretion_Kidneys_Impact For Person 4	Base BGL
1.0	216.315	1.0	221.154
1.0	216.315	1.0	221.154
1.0	216.315	1.0	221.154
1.0	216.315	1.0	221.154
1.0	216.315	1.0	221.154
1.0	216.315	1.0	221.154
Total Time Taken in Seconds	82.009	Total Time Taken in Seconds	49.19

Table 7: PSO Results for Excretion_Kidneys_Impact VS Base BGL for Diabetic Persons

The following Table 8, Table 9 gives the optimized results for the parameter Glycolysis Max Impact and the corresponding result obtained for the target value Base BGL for two normal persons and two diabetic persons

Glycolysis Max Impact_ For Person 1	Base BGL	Glycolysis Max Impact_ For Person 2	Base BGL
0.50266605	90.096	0	69.93
1.0	90.097	0	69.93
0.11100267	90.069	0	69.93
0.74868698	90.097	0	69.93
0.21673699	90.055	0	69.93
0.41997839	90.096	0	69.93
Total Time Taken in Seconds	68.093	Total Time Taken in Seconds	82.908

Table 8: PSO Results for Glycolysis Max Impact VS Base BGL for Normal Persons

Glycolysis Max Impact_ For Person 3	Base BGL	Glycolysis Max Impact_ For Person 4	Base BGL
0.65076068	216.313	0.05193917	220.879
0.83420672	216.314	0	219.739
0.67716967	216.313	0.67609391	221.154
0.71747981	216.314	0.67977369	221.154
0.61424417	216.313	0.84438597	221.153
0.4137003	216.312	0.64155555	221.154
Total Time Taken in Seconds	84.066	Total Time Taken in Seconds	51.148

Table 9: PSO Results for Glycolysis Max Impact VS Base BGL for Diabetic Persons

The following Table 10, Table 11 gives the optimized results for the parameter Glycolysis Min Impact and the corresponding result obtained for the target value Base BGL for two normal persons and two diabetic persons

Glycolysis Min Impact_ For Person 1	Base BGL	Glycolysis Min Impact_ For Person 2	Base BGL
2.6	49.394	2.48	47.371
2.6	49.394	2.48	47.371
2.6	49.394	2.48	47.371
2.6	49.394	2.48	47.371
2.6	49.394	2.48	47.371
2.6	49.394	2.48	47.371
Total Time Taken in Seconds	72.44	Total Time Taken in Seconds	82.941

Table 10: PSO Results for Glycolysis Min Impact VS Base BGL for Normal Persons

Glycolysis Min Impact_ For Person 3	Base BGL	Glycolysis Min Impact_ For Person 4	Base BGL
2.6	153.33	2.6	161.398
2.6	153.33	2.6	161.398
2.6	153.33	2.6	161.398
2.6	153.33	2.6	161.398
2.6	153.33	2.6	161.398
2.6	153.33	2.6	161.398
Total Time Taken in Seconds	82.63	Total Time Taken in Seconds	60.522

Table 11: PSO Results for Glycolysis Min Impact VS Base BGL for Diabetic Persons

The following Table 12, Table 13 gives the optimized results for the parameter Gngimpact and the corresponding result obtained for the target value Base BGL for two normal persons and two diabetic persons

Gngimpact_ For Person 1	Base BGL	Gngimpact_ For Person 2	Base BGL
1.07602018	90.244	1.0	70.256
1.28072756	90.365	1.05609868	70.199
1.01509422	90.203	1.16960161	70.422
1.68485988	90.437	1.0	70.256
1.80299544	90.51	1.0	70.256
1.20764332	90.298	1.03315241	70.252
Total Time Taken in Seconds	82.30	Total Time Taken in Seconds	79.353

Table 12: PSO Results for Gngimpact VS Base BGL for Normal Persons

Gngimpact_ For Person 3	Base BGL	Gngimpact_ For Person 4	Base BGL
1.07602018	90.244	1.23520293	226.07
1.28072756	90.365	1.0	221.154
1.01509422	90.203	1.22361401	226.016
1.68485988	90.437	1.47523068	229.151
1.80299544	90.51	1.59664566	230.302
1.20764332	90.298	1.09205639	224.246
Total Time Taken in Seconds	82.30	Total Time Taken in Seconds	59.28

Liver glycogen breakdown impact_ For Person 1	Base BGL	Liver glycogen breakdown impact_ For Person 2	Base BGL
3.00	90.74	1.55387236	70.426
3.00	90.74	1.85836697	70.176
2.1608399	90.268	3.0	70.556
3.00	90.74	3.0	70.556
1.03977021	90.241	1.14304327	70.447
3.00	90.74	3.0	70.556
Total Time Taken in Seconds	79.172	Total Time Taken in Seconds	82.886

Table 14: PSO Results for Liver glycogen breakdown impact

VS Base BGL for Normal Persons

The above Table 14 and the following Table 15 gives the optimized results for the parameter Liver glycogen breakdown impact and the corresponding result obtained for the target value Base BGL for two normal persons and two diabetic persons

Liver glycogen breakdown impact_ For Person 3	Base BGL	Liver glycogen breakdown impact_ For Person 4	Base BGL
2.04016265	224.71	1.64817374	233.243
1.00	216.315	1.36553881	232.479
2.09629051	224.914	1.15301331	230.699
1.78265385	224.476	2.22368001	232.952
1.11847139	221.845	1.56818727	233.476
1.11038572	221.7	1.8516473	232.498
Total Time Taken in Seconds	89.551	Total Time Taken in Seconds	54.93

Table 15: PSO Results for Liver glycogen breakdown impact VS Base BGL for Diabetic Persons

Liver glycogen synthesis impact_ For Person 1	Base BGL	Liver glycogen synthesis impact_ For Person 2	Base BGL
1.06266536	90.097	1.6248759	70.256
1.6029004	90.097	0.27343285	70.256
0.15922384	90.097	1.86066542	70.256
0.51183072	90.097	1.39346859	70.256
0.63642105	90.097	0.84694322	70.256
0.83930959	90.097	0.19794895	70.256
Total Time Taken in Seconds	87.407	Total Time Taken in Seconds	78.521

Table 16 : PSO Results for Liver glycogen synthesis impact VS Base BGL for normal Persons

The above Table 16 and the following Table 17 gives the optimized results for the parameter Liver glycogen breakdown impact and the corresponding result obtained for the target value Base BGL for two normal persons and two diabetic persons

Liver glycogen synthesis impact_ For Person 3	Base BGL	Liver glycogen synthesis impact_ For Person 4	Base BGL
1.31854762	216.315	1.15006378	221.154
1.74282836	216.315	1.38608173	221.154
0.92724289	216.315	0.43499451	221.154
1.41644504	216.315	1.49906613	221.154
1.54382273	216.315	0.19208556	221.154
0.49920034	216.315	0.17476342	221.154
Total Time Taken in Seconds	76.455	Total Time Taken in Seconds	50.386

Table 17 : PSO Results for Liver glycogen synthesis impact VS Base BGL for Diabetic Persons

The following Table 18, Table 19,Table 20, Table 21 gives the optimized results for all the parameters when they are optimized simultaneously, and the corresponding result obtained for the target value Base BGL for two normal persons and two diabetic persons.

ALL PARAMETERS For Person 1 ["peakInsulinLevel_", "glut4Impact_", "excretionKidneysImpact_", "glycolysisMaxImpact","glycolysisMinImpact_","gngImpact_", "liverGlycogenBreakdownImpact_","liverGlycogenSynthesisImpact_"]	Base BGL
[1.00, 1.00, 1.00, 1.00, 2.6, 2.00, 3.00, 2.00]	91.233
[1.00, 0.48026442, 0.7328305, 0.53927757, 2.22460243, 1.28185344, 1.06039945, 1.6489434]	90.269
[1.00, 1.00, 1.00, 1.00, 2.6, 2.00, 3.00, 2.00]	91.233
[1.00, 1.00, 1.00, 1.00, 2.6, 2.00, 3.00, 2.00]	91.233
[1. 00, 0.43167145, 0.85076422, 1.00, 2.50903026, 1.00, 1.05164298, 1.61682025]	87.764
Total Time Taken in Seconds	114.994

Table 18: PSO Results for All parameters VS Base Insulin Level for Normal Person 1

ALL PARAMETERS for Person 2 ["peakInsulinLevel_", "glut4Impact_", "excretionKidneysImpact_", "glycolysisMaxImpact","glycolysisMinImpact_","gngImpact_", "liverGlycogenBreakdownImpact_","liverGlycogenSynthesisImpact_"]	Base BGL
[1.00, 0.80267567, 0.19140465, 0.26503051, 1.01214847, 1.64748161, 1.00, 0.00]	70.676
[1.0, 1.0, 1.0, 1.0, 2.6, 2.0, 3.0, 2.0]	72.77
[1.0, 1.0, 1.0, 1.0, 2.6, 2.0, 3.0, 2.0]	72.77
[1.0, 1.0, 1.0, 1.0, 2.6, 2.0, 3.0, 2.0]	72.77
[1.0, 1.0, 1.0, 0.3248848, 2.6, 2.0, 2.2013757, 2.0,]	70.901
Total Time Taken in Seconds	100.690

Table 19: PSO Results for All parameters VS Base Insulin Level for Normal Person 2

ALL PARAMETERS for Person 3 ["peakInsulinLevel_", "glut4Impact_", "excretionKidneysImpact_", "glycolysisMaxImpact","glycolysisMinImpact_","gngImpact_", "liverGlycogenBreakdownImpact_","liverGlycogenSynthesisImpact_"]	Base BGL
[1. 0.80267567 0.19140465 0.26503051 1.01214847 1.64748161 1. 0.]	225.791
[1.0, 1.0, 1.0, 1.0, 2.6, 2.0, 3.0, 2.0]	225.791
[1.0, 1.0, 1.0, 1.0, 2.6, 2.0, 3.0, 2.0]	225.791
[0.83744434, 0.44545871, 0.5694835,1 0.11961833, 1.23480395, 1.42902886, 2.5970932, 1.37833681]	225.6
[1. 0, 0.38977069, 0.50943915, 0.70592026, 1.47440696, 1.08025955, 2.14941114, 0.66348723]	225.571
Total Time Taken in Seconds	116.103

Table 20: PSO Results for All parameters VS Base Insulin Level for Diabetic Person 1

ALL PARAMETERS for Person 4 ["peakInsulinLevel_", "glut4Impact_", "excretionKidneysImpact_", "glycolysisMaxImpact","glycolysisMinImpact_","gngImpact_", "liverGlycogenBreakdownImpact_","liverGlycogenSynthesisImpact_"]	Base BGL
[1.0, 1.0, 1.0, 1.0, 2.6, 2.0, 3.0, 2.0]	234.746
[1.0, 1.0, 1.0, 1.0, 2.6, 2.0, 3.0, 2.0]	234.746
[1.0, 1.0, 1.0, 1.0, 2.6, 2.0, 3.0, 2.0]	234.746
[1.0, 0.0, 0.46320527, 0.34247484, 2.19886617, 1.75386451, 1.91659592, 0.5154897]	234.242
[1.0, 1.0, 1.0, 1.0, 2.6, 2.0, 3.0, 2.0]	234.746
Total Time Taken in Seconds	76.8700

Table 21: PSO Results for All parameters VS Base Insulin Level for Diabetic Person 2

Once the results from PSO are completed, then backtracking search optimization algorithm is used. The default parameters used for BSO are as follows:

DEFAULT PARAMETERS	DEFAULT VALUES
Number of Particles	20
mixrate	1
Max_iterations	30
Objective function	run_Simulator
Lower bound	[0,0,0,0,1,1,1,0]
Upper bound	[1,1,1,1,2.6,2,3,2]

Table 22: Default Parameters For BSO

The number of particles for the BSO are kept same as for PSO so that the results obtained from them can be compared and the best result can be selected. Explanation about the remaining parameters were already discussed in the previous section.

The below Table 23 and Table 24 represents the best values for the peak insulin level parameter when trying to optimize the parameter and the corresponding base blood glucose level obtained. The default values in the set-1 from Table 22 were used. four base BGL values were tested with two normal person readings with target base BGL values as 85, 65 and two for diabetic person readings with target base BGL values as 210, 220

Peak Insulin Level For Person 1	Base BGL	Peak Insulin Level For Person 2	Base BGL
0.63045641	93.105	0.71532218	72.233
0.7847314	91.567	0.11377095	110.043
0.41083913	97.522	0.8235889	71.419
0.51611326	94.837	0.44512869	76.55
0.62261828	93.207	0.89195959	70.749
0.8975474	91.097	0.94819649	70.517
Total Time Taken in Seconds	68.6358	Total Time Taken in Seconds	93.6015

Table 23: BSO Results for Peak Insulin Level VS Base BGL for Normal Persons

Peak Insulin Level For Person 3	Base BGL	Peak Insulin Level For Person 4	Base BGL
0.94243926	216.585	0.84182194	221.154
0.59247127	219.478	0.07552246	221.153
0.49397011	219.706	0.35856075	221.153
0.39532866	220.426	0.5585874	221.153
0.95388153	216.529	0.41672234	221.153
0.32538228	220.647	0.9605197	221.153
Total Time Taken in Seconds	74.325	Total Time Taken in Seconds	75.501

Table 24: BSO Results for Peak Insulin Level VS Base BGL for Diabetic Persons

The following Table 25, Table 26 gives the optimized results for the parameter glut4Impact and the corresponding result obtained for the target value Base BGL for two normal persons and two diabetic persons

Glut4Impact_ For Person 1	Base BGL	Glut4Impact_ For Person 2	Base BGL
0.5622236	90.238	0.5458499	70.264
0.1625295	90.336	0.2554479	70.285
0.8878838	90.294	0.011499	70.282
0.2630831	90.325	0.7072489	70.308
0.5449272	90.277	0.3206086	70.349
0.2578622	90.183	0.5938540	70.338
Total Time Taken in Seconds	64.6696	Total Time Taken in Seconds	86.6575

Table 25: BSO Results for Glut4Impact VS Base BGL for Normal Persons

Glut4Impact_ For Person 3	Base BGL	Glut4Impact_ For Person 4	Base BGL
0.33791689	220.438	0. 97413509	221.154
0.48805967	219.698	0. 91728942	221.153
0.3030013	220.599	0. 246709754	221.153
0.70604479	218.601	0. 78809182	221.153
0.10267349	220.598	0. 74656145	221.153
0.57273568	219.627	0. 69836465	221.154
Total Time Taken in Seconds	67.2530	Total Time Taken in Seconds	88.785

Table 26: BSO Results for Glut4Impact VS Base BGL for Diabetic Person

The following Table 27, Table 28 gives the optimized results for the parameter ExcretionKidneysImpact and the corresponding result obtained for the target value Base BGL for two normal persons and two diabetic persons

Excretion_Kidneys_Impact For Person 1	Base BGL	Excretion_Kidneys_Impact For Person 2	Base BGL
0.18096885	90.097	0.26118861	70.256
0.1915564	90.097	0.16241679	70.256
0.92881576	90.097	0.27326758	70.256
0.72136362	90.097	0.0391795	70.256
0.11467464	90.097	0.81664551	70.256
0.61148251	90.097	0.89294145	70.256
Total Time Taken in Seconds	62.8907	Total Time Taken in Seconds	81.251

Table 27: BSO Results for Excretion_Kidneys_Impact VS Base BGL for Normal Persons

Excretion_Kidneys_Impact For Person 3	Base BGL	Excretion_Kidneys_Impact For Person 4	Base BGL
0.38940983	221.277	0.56186546	227.106
0.4425218	221.037	0.00302721	231.467
0.49845855	220.661	0.58969395	226.836
0.94523439	216.771	0.343568	229.736
0.8773414	216.771	0.67875115	226.005
0.03246337	222.082	0.18582175	230.617
Total Time Taken in Seconds	65.343	Total Time Taken in Seconds	89.19

Table 28: BSO Results for Excretion_Kidneys_Impact VS Base BGL for Diabetic Persons

The following Table 29, Table 30 gives the optimized results for the parameter Glycolysis Max Impact and the corresponding result obtained for the target value Base BGL for two normal persons and two diabetic persons

Glycolysis Max Impact_ For Person 1	Base BGL	Glycolysis Max Impact_ For Person 2	Base BGL
0.65815454	90.096	0.3394361	70.255
0.6094395	90.097	0.71569252	70.255
0.98363614	90.069	0.5295703	70.255
0.35445705	90.097	0.53466406	70.255
0.76690436	90.055	0.44092094	70.255
0.61743304	90.096	0.37332785	70.255
Total Time Taken in Seconds	70.994	Total Time Taken in Seconds	67.9721

Table 29: BSO Results for Glycolysis Max Impact VS Base BGL for Normal Persons

Glycolysis Max Impact_ For Person 3	Base BGL	Glycolysis Max Impact_ For Person 4	Base BGL
0.2460428	217.028	0.21007382	220.588
0.71400805	216.314	0.91649989	221.154
0.27060409	216.777	0.613974	221.154
0.75672746	216.314	0.39886162	221.154
0.96789538	216.315	0.5365226	221.153
0.91492068	216.314	0.479396	221.154
Total Time Taken in Seconds	66.528	Total Time Taken in Seconds	97.178

Table 30 : BSO Results for Glycolysis Max Impact VS Base BGL for Diabetic Persons

Glycolysis Min Impact_ For Person 1	Base BGL	Glycolysis Min Impact_ For Person 2	Base BGL
1.2350804	90.034	1.2350804	70.255
1.55232713	89.532	1.55232713	68.255
2.18514348	86.196	2.18514348	72.255
1.89016624	87.666	1.87168754	70.255
1.1494015	86.198	2.12926955	70.255
1.14508933	85.725	2.25173747	70.255
Total Time Taken in Seconds	62.2660	Total Time Taken in Seconds	62.5660

Table 31: BSO Results for Glycolysis Min Impact VS Base BGL for Normal Persons

The above Table 30, Table 31 gives the optimized results for the parameter Glycolysis Min Impact and the corresponding result obtained for the target value Base BGL for two normal persons and two diabetic persons.

Glycolysis Min Impact_ For Person 3	Base BGL	Glycolysis Min Impact_ For Person 4	Base BGL
1.61100046	212.04	1.22611737	220.485
1.91374013	209.362	1.85033173	210.753
2.13807488	203.876	1.87787603	209.641
2.33676765	184.948	1.47584879	219.947
2.26638149	194.294	2.09180469	203.866
1.79370349	210.49	1.08190227	220.955
Total Time Taken in Seconds	64.38	Total Time Taken in Seconds	60.85

Table 32: BSO Results for Glycolysis Min Impact VS Base BGL for Diabetic Persons

The following Table 32, Table 33 gives the optimized results for the parameter Gngimpact and the corresponding result obtained for the target value Base BGL for two normal persons and two diabetic persons

Gngimpact_ For Person 1	Base BGL	Gngimpact_ For Person 2	Base BGL
1.13596161	90.258	1.24683801	70.377
1.34572531	90.306	1.4982994	70.484
1.94268336	90.689	1.86728336	70.588
1.36386042	90.406	1.66719412	70.652
1.29540806	90.329	1.28813649	70.38
1.28665140	90.401	1.00020243	70.256
Total Time Taken in Seconds	68.588	Total Time Taken in Seconds	75.812

Table 33: BSO Results for Gngimpact VS Base BGL for Normal Persons

Gngimpact_ For Person 3	Base BGL	Gngimpact_ For Person 4	Base BGL
1.58950327	222.315	1.64083281	230.236
1.4074984	221.415	1.21210221	226.246
1.96931886	223.315	1.90708991	231.642
1.91335093	223.228	1.08491702	224.398
1.25512664	220.073	1.85595124	231.334
1.22685545	219.46	1.81086172	231.362
Total Time Taken in Seconds	79.082	Total Time Taken in Seconds	67.163

Liver glycogen breakdown impact_ For Person 1	Base BGL	Liver glycogen breakdown impact_ For Person 2	Base BGL
2.92133056	90.974	1.1179864	70.373
1.33333666	90.88	1.4076446	70.639
1.13284268	90.59	2.1996278	70.476
2.31684984	92.49	2.4544109	70.709
1.38185836	90.703	2.7662488	72.282
2.54369610	92.564	2.13396425	70.415
Total Time Taken in Seconds	61.866	Total Time Taken in Seconds	80.6623

Table 35: BSO Results for Liver glycogen breakdown impact

VS Base BGL for Normal Persons

The above Table 34 and the following Table 35 gives the optimized results for the parameter Liver glycogen breakdown impact and the corresponding result obtained for the target value Base BGL for two normal persons and two diabetic persons.

Liver glycogen breakdown impact_ For Person 3	Base BGL	Liver glycogen breakdown impact_ For Person 4	Base BGL
1.49880163	223.956	1.50892756	232.68
1.10626734	221.588	2.36839415	233.905
2.21001199	224.898	1.44381775	232.755
2.17085486	224.825	2.8106602	234.408
2.020817	224.695	1.82751893	233.444
2.98296008	225.529	1.64842339	233.136
Total Time Taken in Seconds	64.4300	Total Time Taken in Seconds	62.651

Table 36: BSO Results for Liver glycogen breakdown impact VS Base BGL for Diabetic Persons

Liver glycogen synthesis impact_ For Person 1	Base BGL	Liver glycogen synthesis impact_ For Person 2	Base BGL
1.86266536	90.097	1.4348759	70.256
0.4019004	90.097	1.17343285	70.256
0.25922384	90.097	0. 59954435	70.256
1.41183072	90.097	0.9812132	70.256
0.73642105	90.097	0.29953322	70.256
0.63730959	90.097	1.70706263	70.256
Total Time Taken in Seconds	91.1500	Total Time Taken in Seconds	86.4083

Table 37: BSO Results for Liver glycogen synthesis impact VS Base BGL for normal Persons

The above Table 36 and the following Table 37 gives the optimized results for the parameter Liver glycogen breakdown impact and the corresponding result obtained for the target value Base BGL for two normal persons and two diabetic persons.

Liver glycogen synthesis impact_ For Person 3	Base BGL	Liver glycogen synthesis impact_ For Person 4	Base BGL
1.81854762	216.315	1.25006378	221.154
0.74282836	216.315	1.08608173	221.154
1.92724289	216.315	0.13499451	221.154
0.21544504	216.315	1.49906613	221.154
0.18660652	216.315	0.09208556	221.154
1.25846948	216.315	0.17476342	221.154
Total Time Taken in Seconds	62.21487	Total Time Taken in Seconds	71.3017

Table 38 : BSO Results for Liver glycogen synthesis impact VS Base BGL for Diabetic Persons

The following Table 38, Table 39,Table 40, Table 41 gives the optimized results for all the parameters when they are optimized simultaneously, and the corresponding result obtained for the target value Base BGL for two normal persons and two diabetic persons.

ALL PARAMETERS for Person 1 ["peakInsulinLevel_", "glut4Impact_", "excretionKidneysImpact_", "glycolysisMaxImpact","glycolysisMinImpact_","gngImpact_", "liverGlycogenBreakdownImpact_","liverGlycogenSynthesisImpact_"]	Base BGL
[0.94254, 0.58480, 0.71906, 0.67798, 2.11756, 1.61109, 1.78030, 0.95920]	90.881
[0.99135, 0.56380, 0.45853, 0.40605, 2.22007, 1.32425, 1.1440, 0.96767]	90.684
[0.96791, 0.75857385, 0.58628911, 0.23586533, 2.15352894, 1.08876751, 1.16764261, 0.90296606]	90.694
[0.98634, 0.96031976, 0.74735705, 0.25968725, 1.21214801, 1.23935211, 1.91508426, 1.14723646]	90.456
[0.980175, 0.7276766, 0.3323185, 0.54250536, 2.35426185, 1.04140811, 1.61993372, 1.55566482]	90.634
Total Time Taken in Seconds	92.155

Table 39: BSO Results for All parameters VS Base Insulin Level for Normal Person 1

ALL PARAMETERS for Person 2 ["peakInsulinLevel_", "glut4Impact_", "excretionKidneysImpact_", "glycolysisMaxImpact","glycolysisMinImpact_","gngImpact_", "liverGlycogenBreakdownImpact_","liverGlycogenSynthesisImpact_"]	Base BGL
[9.78643382e-01, 3.18088777e-01, 4.45493787e-01, 5.27859647e-01, 1.88629497e+00, 1.22351083e+00, 1.99397313e+00, 3.12201057e-01,]	70.56
[9.93035522e-01, 2.41867442e-01, 3.64171981e-01, 6.33956118e-01, 2.34656099e+00, 1.27942521e+00, 1.83020194e+00, 4.51538614e-01]	70.278
[9.59482279e-01, 8.33195338e-01, 7.77417489e-01, 9.20324891e-01, 2.53647130e+00, 1.13251156e+00, 2.33404441e+00, 1.78472220e+00]	70.456
[9.62826678e-01, 2.11538332e-01, 6.00544496e-01, 7.18996412e-01 2.03607768e+00, 1.23125882e+00, 1.82734242e+00, 1.80158187e-01]	70.547
[9.65270790e-01, 4.50256748e-01, 6.84622208e-01, 3.36105549e-01, 2.34437592e+00, 1.06165120e+00, 1.97112233e+00, 1.51435798e+00]	70.448
Total Time Taken in Seconds	93.875

Table 40: BSO Results for All parameters VS Base Insulin Level for Normal Person 2

ALL PARAMETERS for Person 3 ["peakInsulinLevel_", "glut4Impact_", "excretionKidneysImpact_", "glycolysisMaxImpact","glycolysisMinImpact_","gngImpact_", "liverGlycogenBreakdownImpact_","liverGlycogenSynthesisImpact_"]	Base BGL
[0.986640, 0.971946, 0.9172778, 0.7854954, 2.1168844, 1.1776016 1.0645389, 0.89262833]	216.561
[0.980419, 0.7495797, 0.8647664, 0.7924713, 2.0678876, 1.1540305 1.2190351, 0.88072326]	222.995
[0.927892, 0.9631095, 0.92440285, 0.7903968, 2.03728157, 1.2053211, 1.0623311, 0.9873743]	218.411
[0.687966, 0.936486, 0.6948846, 0.81807955, 2.39094365, 1.21073746, 1.0072162, 0.7355212]	213.323
[0.752939, 0.760464, 0.5754805, 0.88900648, 2.25934356, 1.03649003, 1.0155326, 1.3218796]	214.24
Total Time Taken in Seconds	84.20

Table 41: BSO Results for All parameters VS Base Insulin Level for Diabetic Person 1

ALL PARAMETERS for Person 4 ["peakInsulinLevel_", "glut4Impact_", "excretionKidneysImpact_", "glycolysisMaxImpact","glycolysisMinImpact_","gngImpact_", "liverGlycogenBreakdownImpact_","liverGlycogenSynthesisImpact_"]	Base BGL
[0.90235644, 0.77128509, 0.99439871, 0.63518877, 2.1449474 ,1.06722795, 1.04790122, 0.32443425]	216.544
[0.95047154, 0.75134496, 0.22164981, 0.72177402, 2.40014228, 1.03905967, 1.02929895, 0.98930637]	218.866
[0.85813637, 0.90876273, 0.56849679, 0.64790913, 2.02484856, 1.02909449, 1.02163069, 0.29078909]	221.13
[0.93903119, 0.67705747, 0.40288944, 0.71089436, 2.08522135, 1.21630063, 1.02152426, 0.38985107]	229.841
[0.91220444, 0.76790614, 0.46314111, 0.4017865, 1.84763729, 1.18510195 1.0280084,8 1.02211823]	231.751
Total Time Taken in Seconds	86.8700

Table 42: BSO Results for All parameters VS Base Insulin Level for Diabetic Person 2

5. Discussion and Conclusion

In the preceding chapter, we presented the results of the parameters from the particle swarm optimization and backtracking search optimization algorithms. If we take particle swarm optimization algorithm in to consideration the results obtained for all the parameters for normal persons are within the margin of error except for parameter glycolysis_Min_Impact_, here we are obtaining the margin of error as 50 percent which is not acceptable. when we are trying to optimize all the parameters at the same time the best result from the Table 18 is 90.269 for person 1 and 70.676 for person 2 , which are in the acceptable margin of error, But for the two diabetic persons we get the values of about 225.65 and 234.242 which are greater than the margin of error and are not acceptable .The average time taken for single parameter optimization is about 75 seconds and for all the parameters is 100 seconds. There are two problems with PSO: three constants need to be given which are calculated after experimenting with different range of values, which is not efficient. The second problem is that there might be a case when there is a global minimum which is not our target value as in the case of glycolysis_Min_Impact_ which might result in a large margin of error. Hence a more advanced and simpler algorithm backtracking search optimization algorithm was used which would address those issues.

The results produce by the back-tracking search optimization from the previous section are almost similar to the results produced by the brute force method values. The back-tracking search optimization algorithm is efficient and there is only one constant that needs to be given at the starting of the algorithm which is far better than the three constants in the particle swarm optimization algorithm. From the results point of view, it will never stuck at any local minimum and the graphs produced are similar to the brute force method graphs. From the Table 38, Table 39, Table 40 and Table 41, the margin of error is also very small when compared to the particle

swarm optimization algorithm. Therefore, it is better to use the backtracking search optimization algorithm for optimizing the parameters. The results produced from the simulator are within the permissible error limits. Hence the objective of conversion and optimization of the simulator is achieved.

5.1 FUTURE RESEARCH

Since the computing power has increased at a great pace, it is better to venture in to the world of machine learning and deep learning for optimizing the parameters. If there is a good amount of data available, then using the machine learning algorithms like polynomial regression, support vector machines and decision trees will achieve a better result in less amount of time. Since the data is structured, we can also use deep neural networks to train on the data, to achieve the correct results, but using the deep neural networks takes a lot of time and is less recommended.
If better optimization is required then it is recommended to use teacher learner model, learning back tracking search optimization and hybrid back tracking search optimization to achieve the better results, since these are better when the search space increases in the number of dimensions. If there is a need for multi objective optimization the above discussed algorithms can be used with the application of pareto principle, if more efficiency is required for multi objective optimization problems non-dominated ranked genetic algorithms 1 and 2 are recommended which gives the result within the permissible range and are also efficient [6].

From the efficiency point of view when releasing the simulator to the public, it is strictly recommended not to use any optimizing algorithms as they would take a lot of time. It is recommended to use online learning algorithms which will result in better speed, efficiency and less computing power.

REFERENCES

[1] Backtracking Search Algorithm with three constraint handling methods for constrained optimization problems Chunjiang Zhang-Qun Lin-Liang Gao-Xinyu Li - Expert Systems with Applications – 2015

[2] Backtracking Search Optimization Algorithm for numerical optimization problems Pinar Civicioglu - Applied Mathematics and Computation – 2013

[3] A differential invasive weed optimization algorithm for improved global numerical optimization Aniruddha Basak-Dipankar Maity-Swagatam Das - Applied Mathematics and Computation – 2013

[4] Learning backtracking search optimisation algorithm and its application Debao Chen-Feng Zou-Renquan Lu-Peng Wang - Information Sciences – 2017

[5] Backtracking search optimization algorithm based on knowledge learning Debao Chen-Feng Zou-Renquan Lu-Suwen Li - Information Sciences – 2019

[6] Multi-objective backtracking search algorithm for economic emission dispatch problem Mostafa Modiri-Delshad-Nasrudin Rahim - Applied Soft Computing – 2016

[7] A review on simulation-based optimization methods applied to building performance analysis Anh-Tuan Nguyen-Sigrid Reiter-Philippe Rigo - Applied Energy – 2014

[8] Simulation optimization Yolanda Carson-Anu Maria - Proceedings of the 29th conference on Winter simulation - WSC 1997

[9] An Improved Particle Swarm Optimization for Optimal Power Flow Dieu Vo-Peter Schegner - Meta-Heuristics Optimization Algorithms in Engineering, Business, Economics, Finance.

[10] Particle Swarm Optimization Riccardo Poli-James Kennedy-Tim Blackwell - Swarm Intelligence – 2007

[11] An Analysis of Particle Swarm Optimizers, Frans van den Berg, PhD thesis, University of Pretoria – 2001

[12] National Diabetics Statistics Report, Estimates of Diabetes and Its Burden in the United States - 2017