Directional Search Algorithm and its Utilization of Simulated Annealing and Contract-Retract

Beom Jin An
United States Military Academy
Department of Electrical Engineering and
Computer Science
Email: danielan1996@gmail.com

***Abstract*—By building an optimization algorithm, one can understand the details of which factors contribute to creating a sufficient optimization algorithm. In this research, a new algorithm was crafted to experiment on how a combination of existing optimization techniques may improve the performance of the algorithm. The new algorithm, called the Directional Search Algorithm (DSA), utilizes a conglomeration of different techniques typically used in optimization algorithms. These techniques are carefully integrated into one algorithm that geared to finding the global minimum of a single-objective three-dimensional function. DSA is an algorithm derived from an idea that a single search particle extends out in different directions to "scope" out its surroundings. This particle scopes out in multiple directions and finds the solutions at those extended positions. With the incorporation of simulated annealing and a method called "Contraction-Retraction," the particle will be able to explore the search space expansively. Also, the algorithm also offers a unique technique that capitalizes a sufficient balance between exploration and exploitation. The first part of the research will explain and justify the utilization of specific techniques and parameters. The second part of the research will explore the effectiveness of the algorithm. The factors that determine the effectiveness of the algorithm are the following: convergence rate, precision, robustness, and performance. The research examined these four factors on DSA and analyzed the overall performance of the algorithm compared to that of a similar algorithm, the Particle Swarm Optimization (PSO).**

I. Introduction

Optimization problems require an immense amount of computational time as the problem scales increase. It is often unrealistic to compute the solution of an optimization problem using brute force because the problem has a nondeterministic polynomial computational time. Thus, metaheuristic algorithms are built to overcome this computational time problem. Although metaheuristic algorithms do not always output the global optimum, they provide efficiency in approximating optimal solutions. They make assumptions about the problem space to reduce search space and computation.

II. Background

It is often difficult to create an optimization algorithm because of its variety and complex relationships between parameters. Thus, in many cases, optimization algorithms are often inspired by phenomena in nature. The inspired algorithms are nature-proven methods of optimization, and they derive the properties from these natural behaviors that can assist in creating an optimization algorithm that works.

A. Simluated Annealing

One of the most prominent nature-derived methods is "Simulated Annealing." Its inspiration comes from annealing metallurgy, a technique that involves cooling the heated material in a controlled manner to help the molecules form stable crystals. Each material has its optimum cooling rate to create the strongest form of that material after cooling. Similarly, this method can be used to find a global minimum of a system or a problem. The mathematical function used in simulated annealing determines the probability in which the search will accept a worse solution. As the temperature within this function decreases, this probability also decreases. This means that the higher temperature allows for exploration in the initial phase of the search iterations. Once the temperature starts to cool down, the search begins its exploitation and hones down to one area in the search space. Simulated annealing is primarily a tool to balance the exploration and the exploitation within the algorithm. It also offers the probability characteristic that will allow for a more randomized search.

B. Beam Search

In contrast to the simulated annealing method, there is a straight forward concept to pick the solution each step in the optimization algorithm. As simulated annealing offers a probability whether to accept the solution or not, this classic method called "Beam search" is a greedy search algorithm that always accepts the best neighboring node. It explores all possible adjacent nodes and expands to the most promising node. Thus, the probability of accepting a worse solution is always zero. The benefit of the Beam search is that it will approach the solution much quicker than the methods that use probabilities. Also, another advantage is that it does not have any memory requirement. However, the disadvantage of Beam search is that it may converge quicker into a local optimum will not be able to get out. The Beam search does not explore enough of the search space, and its convergence rate is too high. Thus, once the search enters a local minimum, it has no chance of getting out.

C. Particle Swarm Optimization (PSO)

PSO is a metaheuristic that utilizes swarm intelligence to find the optimum solution of a system. Swarm intelligence capitalizes the collective knowledge of multiple particles. The

algorithm initiates with a certain number of particles, and it iteratively improves the positions of these particles by using formulas to determine the position and the velocity of each particle. The particles are guided towards the best solution. The benefit of PSO is that it is more thorough during the exploration process. The particles move across the search space, and they do not "teleport" across the search space. When particles teleport randomly throughout the search space, it loses its exploitative behavior will not be able to settle on a solution. PSO particles converge to the best solution gradually. However, this property can also be a disadvantage as the convergence rate might be too fast before conducting sufficient exploration. PSO does not have a feature that expands its search since it is designed only to converge. Thus, once enough particles swarm into a local minimum, they have no way of exiting that space.

III. Direction Search Algorithm (DSA)

A. Initialization

DSA utilizes multiple agents or particles to share the search space. The algorithm initializes with one particle being placed randomly within the given boundary constraints. The other particles are determined by using the coordinate of the initial particle. DSA calls the initialized particle "the agent," and the other expanded particles "scoping particles." The number of the scoping particles varies depedning on the dimension of the search space. The total $PopSize$ of the particles will be: 3^{dim}. The dim is the "dimension of the scope" which is one less than the dimension of the search space which will be explained in the next subsection. The algorithm makes a matrix called $particles$ that contains the coordinates of the particles which is all initialized with zeroes. This matrix makes it easy for the algorithm to access and update the positions of the particles. Then, an array called fit with the size of the $PopSize$ is created to store the solutions or $fitness$ of the particles. One more additional array is created to store the position of the best particle, in this case, the fitest particle.

B. Directional Movement

As the name describes it, a critical component of DSA is the directional search method. In most cases, the search space is a continuous function. This means that the movement through the search space on a Cartesian coordinate system requires the algorithm to specify the grid coordinates. In contrast, a graph with nodes has a set of adjacent nodes that restricts the movement through the graph. Thus, even within a continuous function, a way of establishing a set of adjacent nodes is required to move through the search space. DSA uses a simple way of choosing these adjacent nodes or in this case, adjacent coordinates. From the starting point of the search space, which is initialized randomly, the adjacent coordinates are determined by adding or subtracting certain "values" from the initial coordinate. (These "values" will be discussed in the next subsection)
[This needs graphical representations]
These adjacent coordinates are called "scoping particles" and the initial or the center coordinate is called the "agent". The scoping particles are on a dimension that is one less than the search space. For example, when searching through a three-dimensional function, you will need two-dimensional scoping particles. One way to look at this is to observe the input variables and the output variable. The particles in the search space are calculating solutions on their respective coordinates. On a three dimensional function, there are two input variables and one output variable. There will always be just one output variable. The two input variables are the dimensions you need to output a solution. Therefore, the particles will have a dimension equal to the number of input variables.

C. Scoping Factor

IV. Results

A single column figure goes here

Fig. 1. Captions go *under* the figure

TABLE I
TABLE CAPTIONS GO *above* THE TABLE

delete	this
example	table

V. Conclusions

bla bla

Appendix A
First appendix

Citation: [1]

Appendix B
Second appendix

Acknowlegment

bla bla

References

[1] N. H. F. Beebe. (2010, Dec.) TEX user group bibliography archive. [Online]. Available: http://www.math.utah.edu/pub/tex/bib/index-table.html