Combined Simulated Annealing and Genetic Algorithm to Solve Optimization Problems

Younis R. Elhaddad

Abstract—Combinatorial optimization problems arise in many scientific and practical applications. Therefore many researchers try to find or improve different methods to solve these problems with high quality results and in less time. Genetic Algorithm (GA) and Simulated Annealing (SA) have been used to solve optimization problems. Both GA and SA search a solution space throughout a sequence of iterative states. However, there are also significant differences between them. The GA mechanism is parallel on a set of solutions and exchanges information using the crossover operation. SA works on a single solution at a time. In this work SA and GA are combined using new technique in order to overcome the disadvantages of both algorithms.

Keywords—Genetic Algorithm, Optimization problems, Simulated Annealing, Traveling Salesman Problem

I. INTRODUCTION

GENETIC ALGORITHM (GA) and Simulated Annealing (SA) have been used to solve optimization problems. Both GA and SA search a solution space throughout a sequence of iterative states. However, there are also significant differences between them. The GA mechanism is parallel on a set of solutions and exchanges information using the crossover operation. SA works on a single solution at a time. GA uses the same selection strategy during the run of the algorithm, while SA regulates the temperature parameter it uses to evaluate the solution. These differences lead to different search criteria. Both GA and SA have advantages and disadvantages. A disadvantage of GA is that it might trap at local minima, or it is time-consuming to find an optimal solution [1]. In case of SA only one candidate solution is used, thus it does not build up an overall view of the search space. Also, SA is slow because of its sequential nature. In other words, SA can find good quality solutions in a neighborhood, but most likely it will get trapped in local minima and takes longer to escape, while GA rapidly discovers the search space, but has difficulty finding the exact minima [2] [3]. The hybridization of SA and GA tries to combine the advantages of GA. In this work GA and SA are combined in order to improve the quality of solutions and reduce execution time.

A. Simulated Annealing

The purpose of physical annealing is to accomplish a low energy state of a solid. This is achieved by melting the solid in a heat bath and gradually lowering the temperature in order to allow the particles of the solid to rearrange themselves in a crystalline lattice structure. This structure corresponds to a minimum energy state for the solid. The initial temperature of the annealing process is the point at which all particles of the solid are randomly arranged within the heat bath. At each temperature, the solid must reach what is known as thermal equilibrium before the cooling can continue [4].

If the temperature is reduced before thermal equilibrium is achieved, a defect will not be frozen into the lattice structure and the resulting crystal will not correspond to a minimum energy state.

The Metropolis Monte Carlo simulation [4] can be used to simulate the annealing method at a fixed temperature $T$. The Metropolis method randomly generates a sequence of states for the solid at the given temperature. A solid’s state is characterized by the positions of its particles. A new state is generated by small movements of randomly chosen particles. The change in energy $\Delta E$ caused by the move is calculated and acceptance or rejection of the new state as the next state in the sequence is determined according to Metropolis acceptance condition [4]. If $\Delta E < 0$ the move is acceptable and if $\Delta E > 0$ the move is acceptable with probability, if $e^{-\frac{\Delta E}{T}} > \Omega$.

The move is acceptable otherwise rejected, where $\Omega$ is random number and $0 < \Omega < 1$. Simulated annealing was first introduced by Metropolis et al [5], but it was Kirkpatrick et al [6], in 1983 who proposed SA as the basis of an optimization technique for combinatorial optimization problems. Simulated annealing is one of the most popular and general adaptive heuristic algorithms [7]. Simulated annealing algorithms have been applied to solve numerous combinatorial optimization problems. The name and idea of SA comes from annealing in metallurgy, a technique involving heating and controlled cooling of a material to increase the size of its crystals and reduce their defects. The heat frees the atoms to move from their initial positions (initial energy). By slowly cooling the atoms the material continuously rearranges, moving toward a lower energy level. They gradually lose mobility due to the cooling, and as the temperature is reduced the atoms tend to crystallize into a solid. In the simulated annealing method, each solution $S$ in the search space is equivalent to a state of a physical system and the function $f(S)$ to be minimized is equivalent to the internal energy of that state. The objective is to minimize the internal energy as much as possible. For successful annealing it is important to use a good annealing schedule, where the temperature reducing gradually.

B. Annealing schedule

The annealing schedule is a major step of SA because it controls the uphill movement of the algorithm. In order to use simulated annealing, the annealing schedule must be at a proper setting. The initial temperature is set at a high value and then it is decided how to decrease the temperature gradually as a function of time. If $T(0)$ is the initial temperature, a simple schedule is $T(k+1) = \beta T(k)$, for some fixed $\beta$, such that $0 < \beta < 1$. A cooling schedule depends on the problem to be solved.
Choosing an annealing schedule for a given problem is still a problem for most researchers. Therefore the annealing schedule can be determined by experiments or using heuristics methods to find the optimal annealing parameters.

**C. Description of SA**

As shown in figure (1) SA starts from a random solution \( x_p \), selects a neighboring solution \( x_n \) and computes the difference in the objective function values, \( \Delta f = f(x_n) - f(x_p) \). If the objective function is better (\( \Delta f < 0 \)), then the present solution \( x_p \) is replaced by the new one \( x_n \).

Otherwise the solution that decreases the value of the objective function with a probability \( pr = 1/(1 + e^{-\Delta f/T}) \) is accepted, where \( pr \) is decreased as the algorithm progresses, and where (\( T \)) is the temperature or control parameter. This acceptance is achieved by generating a random number (\( rm \)) where (0 ≤ \( rm \) ≤ 1) and comparing it against the threshold. If \( pr > rm \) then the current solution is replaced by the new one. The procedure is repeated until a termination condition is satisfied.

---

**Abstract Genetic Algorithm (AGA).**

BEGIN GA
Create initial population randomly
While NOT stop DO
BEGIN
Select parents from the population.
Produce children from the selected parents.
Mutate the individuals.
New population = children
END
Output the best individual found.
END GA

**Fig. 2 Abstract Genetic**

**III. PRINCIPLE OF SAGA**

In this work two SA algorithms are used to generate new two individuals which are transferred to a genetic algorithm, using multi crossover technique and swapped inverted crossover [10] to produce offspring consist of 94 children, from these offspring the best two individuals are selected according to their fitness values, once again these two individuals are used as inputs to both of the SA algorithms in order to improve these solutions. If solutions are no longer improved within consecutive iterations, then the best memorized solutions from the SA algorithms will be moved to the GA to repeat the above process.

The role of the SA is to improve or change the population which caused the GA to become trapped at local minima. The algorithm presented in this work can be effective to solve TSP with different sizes and tend to produce better quality results in lesser time.