3.1.4 Comparison Genetic Algorithms with Simulated Annealing

Simulated annealing is a well-known, high-performance optimization technique for combinatorial problems. The simulated annealing algorithm is presented in Fig.3.7. The temperature is initialized to a relatively high value, and it is slowly decreased until a "freezing point" is reached. At each temperature, components are selected for possible movement until equilibrium is reached. If movement of the selected components results in an improved placement, the movement is performed. Otherwise, the movement is performed with a probability that decreases exponentially with temperature. Components are typically selected randomly for pairwise exchanges.

```
Temperature = InitialTemperature;
currentPlacement = randomInitialPlacement:
currentScore = score (currentPlacement);
While equilibrium at Temperature not reached Do
    selectedComponents = select (atRandom);
    trialPlacement = move (selectedComponents, atRandom);
    trialScore = score (trialPlacement);
    If trialScore < currentScore then
        currentScore = trialScore;
        currentPlacement = trialPlacement;
    Else
        If uniformRandom (0, 1) <
        e^-(trialScore - currentScore)/Temperature then
            currentScore = trialScore;
            currentPlacement = trialPlacement;
    Temperature = Temperature × Alpha;   // Alpha ≈ 0.95
```

Figure 3.7. Simulated annealing algorithm

Both simulated annealing and the genetic algorithm are computation intensive. However, the genetic algorithm has some intrinsic features which, if exploited properly, can result in significant savings. One difference ft that simulated annealing operates on only one solution at a time, while the genetic algorithm maintains a large population of solutions which are optimized simultaneously. Thus the genetic algorithm takes advantage of the experience gained in past exploration of the solution space, and it can direct a more extensive search to areas of lower average cost. Since simulated annealing operates on only one solution at a time, it has very little history to use in learning from past trials.

Both simulated annealing and the genetic algorithm have mechanisms for avoiding entrapment at local optima. In simulated annealing, this is accomplished by occasionally discarding a superior solution and accepting an inferior one. The genetic algorithm also relies on inferior individuals as a means of avoiding false optima, but. since it has a whole population of individuals, the genetic algorithm can keep and process inferior individuals without losing the best ones. Furthermore, in the genetic algorithm, each new individual is constructed from two previous individuals, which means that in a few iterations, all the individuals in the population have a chance of contributing their good features to form one super-individual. In simulated annealing, each new solution is formed from only one old solution, which means that the good features of radically different solutions never mix. A solution is either accepted or thrown away as a whole, depending on its total cost.

Simulated annealing is an inherently serial algorithm. We are looking for an algorithm that can be efficiently parallelized on distributed computers, such as a workstation network connected by Ethernet. Such workstation networks, unlike specialized parallel hardware, are very popular today. The genetic algorithm can be parallelized on such loosely coupled distributed computer networks with 100% processor utilization [148]. Substantial research has been done to parallelize simulated annealing [27], [62]. Although some moderately efficient parallel implementations of simulated annealing exist for shared memory machines, it is known to be a serial algorithm that cannot be efficiently parallelized in the distributed workstation environment.