Baltic J. Modern Computing, Vol. 7 (2019), No. 3, 436-443
https://doi.org/10.22364/bjmc.2019.7.3.10

Study of Convergence in Metaheuristics Algorithms

Donatas KAVALIAUSKAS[1], Leonidas SAKALAUSKAS[2]

[1]Data Science and Digital Technologies, Vilnius university, Vilnius, Lithuania
[2]Fundamental Sciences, Vilnius Gediminas technical university, Vilnius, Lithuania

donatas.worshipper@gmail.com, leonidas.sakalauskas@vgtu.lt

Abstract. Artificial intelligence (AI) system purpose is to help humans solve problems. This branch of science became famous less than a hundred years ago. Since then, it has gained momentum and scale. This area is currently associated with many methodologies, some of which are called metaheuristics algorithms. In this work, we will look at several metaheuristics algorithms. Comparison of algorithm solutions will be performed. We compare the accuracy of the results, the speed of the solution, and other parameters. They will solve one of the classic NP problems. This problem is named a scheduling problem. This paper presents an approach for enhancement of this balance in single solution metaheuristics applied to solve two processors scheduling problem generated during metaheuristic search. We compare Simulated Annealing (SA) algorithm with our develop modification amongst to other well-known metaheuristics like a genetic algorithm (GA) and artificial ant colonies algorithm (ACA) taken from the source of literature.

Keywords: Artificial intelligence, metaheuristics,

Introduction

Many discrete problems are NP-hard in nature and therefore are possible to solve only by exponential or higher complexity algorithms. To this end, the attention is drawn on heuristic or metaheuristic methods to allow us to address the discrete global optimization problems effectively. By contrast, the variables in discrete optimization problems are allowed to take on any values permitted by the constraints.

A successful metaheuristic should provide a balance between the exploration (diversification) and the exploitation (intensification) of the search space. An investigation is needed to identify parts of the search space with high-quality solutions to intensify the search in some promising areas of the accumulated search experience. Of course, enhancement of a balance between exploration and exploitation is a challenging task for improving the efficiency of metaheuristics. This paper presents an approach for enhancement of this balance in single solution metaheuristics applied to solve discrete problems, varying the neighborhood of solution generated during metaheuristic search. The modification of the well-known metaheuristics of genetic algorithm (GA), artificial ant colonies algorithm (ACA) and Simulated Annealing (SA) algorithm with modification. These algorithms are widely applied for solving various discrete global optimization problems.

Simulated Annealing algorithm

Simulated annealing is a global search and optimization method, that is based on the principles of thermodynamics, and from the statistical standpoint, it ensures achieving global optimum if temperature decrease rate (temperature schedule) is chosen properly (Gelatt et al., 1983). This method has extensive functionality in both continuous and discrete regions, simultaneously. SA choose a point with a density function g(x) around current state x*. Then, the cost of new state is calculated by cost function, and at last, an acceptance function h(x) decides to accept this state or not. It is clear that if the new rule has lower cost, it has a higher chance to be taken. At the beginning of the search, range of possible new points is extensive, and the possibility of acceptance and denial is almost equal due to high temperature. With decrease in temperature, a variety of new state will be decreased, which leads to an increase in resolution.

In the next part, we will explain how SA works. Our adapted SA algorithm for discrete global optimization can be described as follows.

Step 0. Let x_0 ε X be a given starting point, $z_0 = \{x_0\}$ and k = 0;

Step 1. We calculate the temperature parameter T at iteration k on which the number of elementary transformation operations depends. We constructed special neighborhood depth (ρ_k) generating algorithm, based on generation of stable Pareto values:

Step 1. 1. Set initial $i := 1$ and $T := 0$.

Step 1. 2. Generate *U1* and *U2*, uniformly distributed in (Gelatt et al., 1983), and (Weise, 2011)**.** Then calculate:

$$Z := \left(\frac{sin((\alpha-1)\cdot U1\cdot\pi)}{-\,ln(U2)}\right)^{\frac{1-a}{a}} \cdot \frac{\cdot sin(\alpha\cdot U1\cdot\pi)}{\left(cos(\alpha\cdot\frac{\pi}{2})\cdot sin(U1\cdot\pi)\right)^{\frac{1}{\alpha}}} \tag{1}$$

$$T := T + Z \tag{2}$$

where the value of parameter α is selectable between 0 and 1. In the following calculation value of this parameter was used as 0.85.

Step 1. 3. Calculating a ρ_k value:

$$\rho_k = \begin{cases} i, & T \geq T_0 \\ i+1, & T < T_0 \end{cases} \tag{3}$$

here neighborhood depth (ρ_k) value is monotonically depends on temperature and parameter *i* belongs to the number of iterations. The parameter T_0 has a predefined value that is half the number of possible allowable elementary transforms. In terms of assigning a task to one executive or another, that would be half the number of tasks available. (Gelatt et al., 1983).

Step 2. A new solution is being developed by carrying out elementary rearrangements up to the permitted temperature by volume to the solution set In the following calculations as simply rearrangements were used randomly shift and swap operation in the solution set.

Step 3. We calculate the objective function with new meanings of values.

Step 4. The resulting value is checked using the Metropolis-Hastings (MH) criterion algorithm, which is expressed in:

Step 4.1. Select the initial value θ;

Step 4.2. Iteration t proposes to go to the value of θ * with the probability Jt (θ * | θ (t-1));

Step 4.3. Calculate the acceptable ratio (opportunity) (Weise, 2011):

$$r = \frac{p(\theta^0|y)/J_t(\theta^*|\theta^{(t-1)})}{p(\theta^{(t-1)}|y)/J_t(\theta^{(t-1)}|\theta^*)} \tag{4}$$

Step 4.4. We accept θ * as θ (t) with the option min (r, 1). If θ * is not accepted, then θ (t) = θ (t-1).

Step 5. Check a stopping criterion and if it fails set k = k + 1 and go back to **Step 1.**

Genetic algorithm

Genetic algorithms are adaptive and can be used to solve search and optimization tasks. This algorithm was created based on genetic processes occurring in nature. Many times, the natural population evolves based on the principle of natural selection and the policy of the survival of the most influential individual, described for the first time in Charles Darwin's Book of Kinds. By mimicking this process, genetic algorithms can adapt to real-life situations and find their solutions if the algorithm is properly designed (Wang, 2017).

Before starting to solve the problem, it is necessary to find the right coding (or representation) of the problem to be addressed. There is also a need for a verification function that allows us to assess the quality of certain individuals (or just solutions) to make it possible for two individuals to compare with each other. During the implementation of the realization of the genetic algorithm, two parents who cross each other to obtain the offspring are constantly selected. Let the task solutions are presented in parameter sets. Individuals in one iteration (once) are selected and crossed and thus produce offspring within the next population (iteration). Parents are randomly selected from the population according to a scheme that favors better individuals. When two parents are chosen, their chromosomes are combined using crossover and mutation procedures. The mutation, which is a random chance of genes, is used to maintain population diversity (Felinskas et al., 2006).

GA algorithm for discrete global optimization can be described as follows.

Step 0. Randomly generate chromosome solution and calculate their goal function meaning.

Step 1. Evaluate each chromosome solution and mark the best solution. A chromosome is encoding as one array, where can be value be 0 or 1. The 0 value means that the job is taken another machine and 1 represents that the job is taken this machine.

Step 2. Randomly select another chromosome which will do a crossover method, with chromosome which gives the best solution. Crossover operation we choose to use 2-point crossover method where is randomly selected crossover points. This method is illustrated in Figure 1.

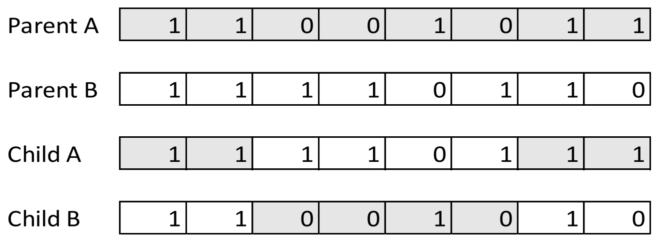

Figure 1. 2-point crossover example.

Step 3. Mutation method. This method is used to mutate best solution chromosome. Randomly are changed the best solution chromosome cells from 0 to 50%.

Step 4. Attributed new solutions to chromosomes and calculate theirs new goal function meaning.

Step 5. Check a stopping criterion and if it fails to go back to **Step 1.** (Weise, 2011)**.**

Artificial ant colonies algorithm

Ant colonies algorithm is based on seeking food. This algorithm is based on real ants observation. Then ants start looking for food they are looking it randomly, but then a portion of food is found ants leave strong pheromone smell to other ants, that a food resource find quickly, and a journey distance be shorter. The pheromone on the shorter path will, therefore, be more strongly reinforced and will eventually become the preferred route for the stream of ants. (Weise, 2011) and (Bianchi et al., 2002).

ACA algorithm for discrete global optimization can be described as follows.

Step 0. All ants get the same primary solution and goal function meaning.

Step 1. If it is the first iteration, we randomly generate a solution for ants and calculate their goal function meaning. Otherwise the solution of each individual is chosen according to the probability attributed to the edges of the matrix G (i, j), but also the heuristical distance corresponding to the edge (i, j) is used for the decision, here the probability of the transition from i to j to the kth ant at time t is defined (Simonavičius, 2007),

$$P^{k}{}_{i,j} = \begin{cases} \frac{[\tau_{i,j}(t)]^{\alpha}[\eta_{i,j}]^{\beta}}{\Sigma[\tau_{i,k}(t)]^{\alpha}[\eta_{i,k}]^{\beta}} if(i,j) \notin tabu_k \\ k \in \text{allowed}_k \\ 0 \end{cases} \quad (5)$$

where is the set of prohibited transitions, o. The parameters α and β are determined by the expert, evaluating the probability and the influence of the distances on the decision. Once set, only distance information will be used, and vice versa.

Step 2. Evaluation all ants new goal function meaning.

Step 3. Updating each ant local pheromone matrix by the formula (Bianchi et al., 2002):

$$\Delta\tau^{k}{}_{i,j} = \begin{cases} \frac{Q}{L_k} if\ k\ ant\ goes\ (i,j) edge \\ 0 \end{cases} \quad (6)$$

where is the kth individual path length between time t and t + n, Q is a positive constant. Thus, the probability of Q / L is recalculated in each iteration.

Step 4. Recalculated global pheromone matrix. The probability of the individual left behind is calculated according to the principle of belated renewal. It is expressed by function (Weise, 2011):

$$\tau_{i,j}(t+n) = \rho \cdot \tau_{i,j} + \Delta\tau_{i,j} \quad (7)$$

where ρ is the coefficient chosen to represent the loss of probability (pheromone evaporation factor) by the edge (i, j) between the time t and t + n. In order for the probabilities not to grow too fast, it should be valid. All probabilities obtained by m ant are calculated as follows (Simonavičius, 2007):

$$\Delta\tau_{i,j} = (1-p) * \tau_{i,j} + \sum_{k=1}^{m} \Delta\tau^{k}{}_{i,j} \quad (8)$$

Step 5. Check a stopping criterion and if it fails go back to **Step 1.**

Computer modeling

We made some computational research, computer modeling of efficiency of this algorithm. In global optimization problems, when we used some optimization algorithms, we need to test the reliability and efficiency of these algorithms. We can use some special testing functions, well known in the literature. Some of these functions have one or more global minimum; some of them have global and local minimums. With the help of these functions, we can ensure, that our methods are efficient enough, we can test and prevent algorithms from being trapped in a local minimum, we can watch the speed and accuracy of convergence and other parameters.

Formulation of the Problem

We have two processors, which share jobs. There is a 100-job list. Each job duration is randomly generated by Gaussian distribution. If the first processor takes a job from the list, then the second processor can no longer take the same job. All jobs must be assigned to processors so that theirs all duration time is shorter. It should be mentioned that the processors are of equal capacity. All testing algorithms repeated calculations 2000 times.

Results

In this section, we will compare algorithms by their goal function meaning, convergence speed and showing similarity to Weibull cumulative distribution.

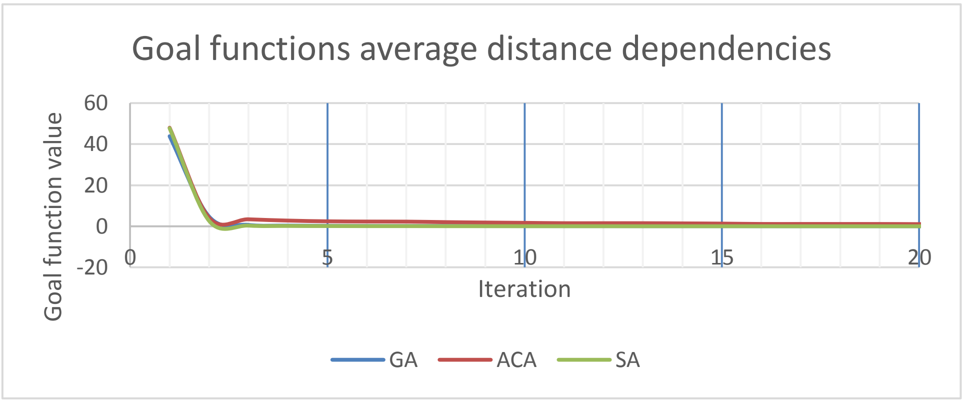

Figure 2. Goal functions average distance dependencies.

The Figure 2 shows, that SA algorithm goal function is approaching the optimal solution in the first iterations compering with other algorithms. Between 1 and 3 iterations we see dramatically goal function alteration. Other iterations only reduce the e change to the optimal solution.

The Figure 3 shows that algorithms convergence speed is directly proportional to the number of iterations. This algorithm is on a logarithmic scale, that helps more clearly to see algorithms convergence speed in a whole calculation life cycle. According to this graph we can see that ACA algorithm was stuck in local optimum, although this algorithm gave good results according to Figure 2 graph.

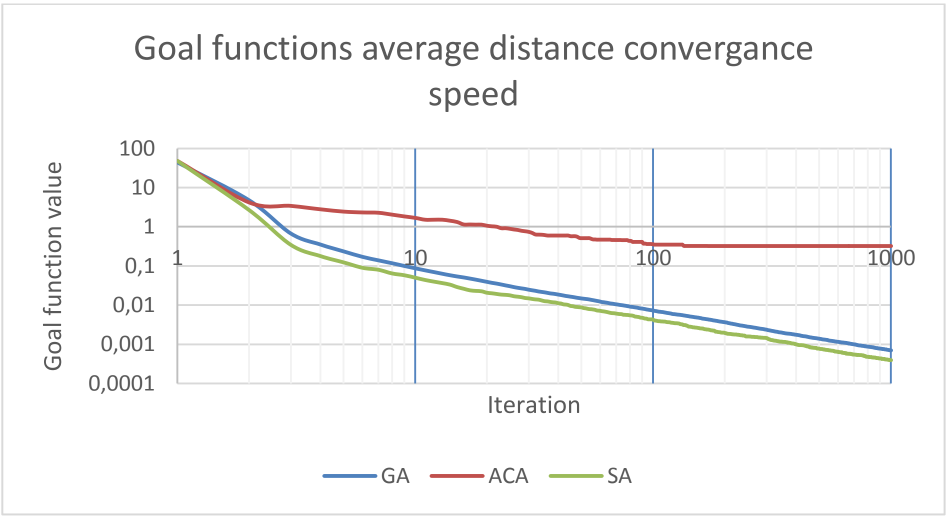

Figure 3. Goal functions average distance dependencies.

The Figure 3 shows that algorithms convergence speed is directly proportional to the number of iterations. This algorithm is on a logarithmic scale, that helps more clearly to see algorithms convergence speed in a whole calculation life cycle. According to this graph we can see that ACA algorithm was stuck in local optimum, although this algorithm gave good results according to Figure 2 graph.

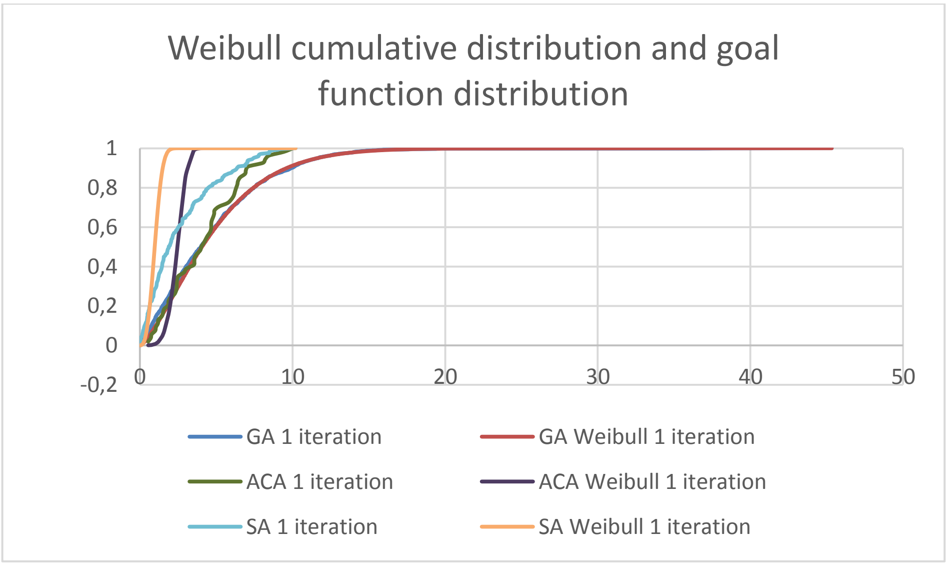

Figure 4. Weibull cumulative distribution and goal function distribution.

The Figure 4 shows that algorithms goal functions are very similar to Weibull cumulative distribution, notably GA algorithm. This indicates that we can use Weibull distribution rules on our solutions. In other iterations, all algorithms goal function distributions are more similar with Weibull cumulative distributions. In this paper, we show first iteration results.

Conclusions

Since the use of metaheuristics algorithms for the optimal solution requires a lot of computer time and operations, the main objective functional improvement is achieved already during the first iterations of the operation. Therefore, it is advisable to use heuristic algorithms with a small iterative step. Our develop SA algorithm is more effective in first iteration compering with other algorithms in this paper. This shows the algorithm's ability to solve more complex problems.

The efficiency of the algorithm depends on the selection of its parameters. Therefore, the selection and application of its settings for the class of tasks to be addressed must be implemented. ACA algorithm letting randomly selecting α and β parameters could be a possibility that this algorithm shows the worse solution in this paper.

Furthermore, our developed metaheuristic algorithm efficiency research methodology for continuous problems is applicable to discrete issues. This methodology let lets you investigate algorithms convergence speed and efficiency, thus complementing standard statistical methods.

References

Bianchi, L., Gambardella, L.M., Dorigo, M. (2002). An ant colony optimization approach to the probabilistic traveling salesman problem. PPSN-VII. Seventh International Conference on Parallel Problem Solving from Nature. Lecture Notes in Computer Science, Springer Verlag.

Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P. (1983). Optimization by Simulated Annealing. Science, New Series. Vol. 220, No. 4598, 671-680.

Sakalauskas, L., Felinskas, G. (2006). Optimization of resource constrained project schedules by genetic algorithm based on the job priority list. Information technology and control. Vol 3, No 4, 412-418, available at: ttp://itc.ktu.lt/index.php/ITC/article/view/11813/6480

Simonavičius, J. (2007). Gamybinių tvarkaraščių sudarymo uždavinio algoritmai ir analizė. Master thesis, University of Kaunas technology, Lithuania, available at: *http://talpykla.elaba.lt/elaba-f*edora/objects/elaba:1855149/datastreams/ATTACHMENT_1855152/content

Wang, J. (2017). Optimization and Simulation of Resource Constrained Scheduling Problem Using Genetic Algorithm. Science journal of business and management, Vol 4, No 6, 229-237, available at http://article.sciencepublishinggroup.com/pdf/10.11648.j.sjbm.20160406.18.pdf

Weise, T. (2011). Global Optimization Algorithms –Theory and Application, 3rd ed., available at http://www.it-weise.de/projects/bookNew.pdf

Received September 8, 2019, accepted September 25, 2019