Utilization of Modified Simulating Annealing as a Tool for Parallel Computing

JAN PANUŠ
Faculty of Economics and Administration
University of Pardubice
Studentska 95, Pardubic 53210, Czech republic
CZECH REPUBLIC
jan.panus@upce.cz

Abstract: The paper deals with using modification of simulated annealing as a part for parallel computing of some typical optimization problems. We will test modified algorithm on set of testing function. The algorithm is created with knowledge taken from basic local search algorithm, tabu search and simulated annealing.

Key-Words: Parallel computing, travelling salesman problem, optimization, simulated annealing, testing functions

1 Introduction
Parallel computing is a strong tool for large datasets [13] or large defined problems computing. It can be used in many areas of global society activities such as business, government, science, research or entertainment [2],[7]. Computer machines with more than one processor or connected together are common platforms in these days. It is very important to understand the principles, techniques and programming techniques for parallel computing to have lower costs, better performance and usable results. Development of parallel software has traditionally been thought of as time and effort intensive. This can be largely attributed to the inherent complexity of specifying and coordinating concurrent tasks, a lack of portable algorithms, standardized environments, and software development toolkits. It is clear that architecture with just one processor is not sustainable performance in the future. Parallel computing is the future of making applications for research, entertainment or business.

One part of parallel computing is based on optimization theory. Parallel computing is valuable tool for finding best solution in large areas of defined problems. What usually took huge amount of time or processor’s possibility to compute the problem in the past, it takes pleasing shortening of time. We will present a part from optimization theory which is based on evolutionary theory. There are many types of algorithm to use for optimization and parallel computing, typical one is genetic algorithm [14]. But it is possible to use different ones. We use the set of optimization algorithms of stochastic type of algorithm. One is simulated annealing [1],[12], another is local search algorithm [4] and last one is tabu search [5],[6]. We take these three algorithms and make some modification that will be explained in the paper. These algorithms are typical optimization algorithms and usually have reasonable results. Utilization of combination of these algorithms and parallel computing is possible in such science disciplines as computational physics or chemistry. It is possible to use it for designing of new materials, weather modeling, floods predictions etc. Connection between research and day-to-day life problems is obvious.

Other utilization of parallel algorithm is in graph theory [10] that plays significant role in computer science because it provides systematic way to model many problems. We will focus on testing our proposal on some testing functions [3]. These testing functions have good ability to test every optimization algorithm. Results obtained from these calculations are possible to transform to graph theory problems. It is possible to say whether results will be satisfactory even for graph theory problems. We would to focus on some graph theory problem in the future research.

We will introduce the modification of simulated annealing problem in the next chapter.
2 Modified Simulated Annealing Algorithm

2.1 Solution Features

In this section, we briefly overview simulated annealing (SA) and its theory [1] for solving discrete unconstrained nonlinear programming problems (NLP) or combinatorial optimization problems. A general unconstrained NLP is defined as: minimize $f(i)$ for $i \in S$

1. procedure SA
2. set starting point $i = i_0$;
3. set starting temperature $T = T_0$ and cooling rate $0 < \alpha < 1$;
4. set $NT$ (number of trials per temperature);
5. while stopping condition is not satisfied do
   for $k \leftarrow 1$ to $NT$ do
5.1 generate trial point $i'$ from $S_i$ using $q(i, i')$;
5.2 accept $i'$ with probability $AT(i, i')$
6. reduce temperature by $T \leftarrow \alpha \times T$;
7. end_while
8. end_procedure

Fig. 1 - Simulated annealing (SA) algorithm taken from [1]

where $f(i)$ is an objective function to be minimized, and $S$ is the solution space denoting the finite set of all possible solutions.

A solution $i_{opt}$ is called a global minimum if it satisfies $f(i_{opt}) \leq f(i)$, for all $i \in S$. Let $S_{opt}$ be the set of all the global minima and $f_{opt} = f(i_{opt})$ be their objective value. Neighborhood $S_i$ of solution $i$ is the set of discrete points $j$ satisfying $j \in S_i \leftrightarrow i \in S_j$. Figure 1 shows the procedure of SA for solving unconstrained problem (1) $q(i; i')$, the generation probability, is defined as $q(i; i') = 1/|S_i|$ for all $i' \in S_i$, and $AT(i; i')$, the acceptance probability of accepting solution point $i'$, is defined by:

$$A_T(i, i') = \exp\left(-\frac{f(i') - f(i)}{T}\right),$$

(1)

where $a^+ = a$ if $a > 0$, and $a^+ = 0$ otherwise.

Accordingly, SA works as follows. Given current solution $i$, SA first generates trial point $i'$. If $f(i') < f(i)$, $i'$ is accepted as a starting point for the next iteration; otherwise, solution $i'$ is accepted with probability $\exp\left(-\frac{f(i') - f(i)}{T}\right)$. The worse the $i'$ is, the smaller is the probability that $i'$ is accepted for the next iteration. The above procedure is repeated $NT$ times until temperature $T$ is reduced. Theoretically, if $T$ is reduced sufficiently slowly in logarithmic scale, then SA will converge asymptotically to an optimal solution $i_{opt} \in S_{opt}$ [1].

In practice, a geometric cooling schedule, $T \leftarrow \alpha T$, is generally utilized to have SA settle down at some solution $i^*$ in a finite amount of time. SA can be modeled by an inhomogeneous Markov chain that consists of a sequence of homogeneous Markov chains of finite length, each at a specific temperature in a given temperature schedule. According to generation probability $q(i; i')$ and acceptance probability $AT(i; i')$, the one-step transition probability of the Markov chain is:

$$P_T(i, i') = \begin{cases} q(i, i')A_T(i, i') & \text{if } i \in S_i \\
1 - \sum_{j \in S_i,i \neq i} P_T(i, j) & \text{if } i' = i \\
0 & \text{otherwise} \end{cases}$$

(2)

and the corresponding transition matrix is $P_T = \{P_T(i, i')\}$.

It is assumed that, by choosing neighborhood $S_i$ properly, the Markov chain is irreducible [1], meaning that for each pair of solutions $i$ and $j$, there is a positive probability of reaching $j$ from $i$ in a finite number of steps.

Consider the sequence of temperatures $\{T_k; k = 0; 1; 2; \ldots\}$, where $T_k > T_{k+1}$ and $\lim_{k \to \infty} T_k = 0$, and choose $N_T$ to be the maximum of the minimum number of steps required to reach an $i_{opt}$ from every $j \in S$. Since the Markov is irreducible and search space $S$ is finite, such $N_T$ always exists. The asymptotic convergence theorem of SA is stated as follows [1].

The Markov chain modeling SA converges asymptotically to a global minimum of $S_{opt}$ [1] if the sequence of temperatures satisfies:

$$T_k \geq \frac{N_T \Delta}{\log_e(k + 1)},$$

(3)

where $\Delta = \max_{j \in S} \{f(j) - f(i) \mid j \in S_j\}$.

The proof of this theorem is based on so-called local balance equation [1], meaning that:

$$\pi_T(i)P_T(i, i') = \pi_T(i')P_T(i', i),$$

(4)
where $\pi_T(i)$ is the stationary probability of state $i$ at temperature $T$.

Although SA works well for solving unconstrained NLPs, it cannot be used directly to solve constrained NLPs that have a set of constraints to be satisfied, in addition to minimizing the objective. The widely used strategy is to transform constrained NLP into an unconstrained NLP using penalty formulation. For static penalty formulation, it is very difficult to choose suitable penalty: if the penalty is too large, SA tends to find feasible solutions rather than optimal solutions. For dynamic penalty formulation, unconstrained problem at every stage of $\lambda(k)$ has to be solved optimally in order to have asymptotic convergence. However, this requirement is difficult to achieve in practice, given only a finite amount of time in each stage. If the result in one stage is not a global minimum, then the process cannot be guaranteed to find constrained global minima. Therefore, applying SA to a dynamic-penalty formulation does not always lead to asymptotic convergence. Besides, SA cannot be used to search in a Lagrangian space, because minimizing Lagrangian function (Panus and Simonova, 2005) does not guarantee constraint satisfaction.

2.2 Penalty Modifications

If algorithm find a local minimum $s^*$, the penalty modification started. Penalty parameter is incremented by utility expression:

$$\text{using} (s^*, f_i) = U_i(s^*) \times \frac{c_i}{1 + p_i}. \quad (5)$$

We use penalty parameter for not stick in local minima. Each parameter is increasing by defined value until the increased value of function is bigger than value of neighbor. The role of penalty parameter in eq. (5) is to count how many times a feature has been penalized.

2.3 Regularization Parameter

Important parameter for local search and for simulated annealing is regularization parameter that impresses behavior of algorithm. We called this parameter $\lambda$. This parameter determines the degree up to which constraints on features are going to affect searching in the defined problem. We test how this parameter is going to affect the moves performed by a simulated annealing. The range of the parameter is shown in eq. 6.

3 Testing

We tested the value of $\lambda$ parameter at every instances of every problem. The value of this parameter varied as reader can see below:

For testing functions was measured following values:

a) number of iterations necessary for achievement of optimal solution (or at least sub-optimal),

b) the best found solution,

c) number of change of the best found solutions – it means how many times we measured change of the best found solution to next better solution in local search procedure

d) time spend on finding best found solution on PC computer with Intel Core2 Duo, 2.66 GHz and 1.96 GB RAM memory. We used many test functions from [3]. We used Sphere model – $1^{st}$ de Jong function (7) and Griewangk’s function (8).

$$\sum_{i=1}^{n} x_i^2; -5.12 \leq x_i \leq 5.12 \quad (7)$$

We can see that modified algorithm can find optimum for every value of $\lambda$ pretty good (fig.3) and pretty fast (fig. 2). If we want to tune the parameter best results are obtain within range of <0.3; 0.5>. Number of best found (or accepted) solutions (fig. 4) means how many time the algorithm changed previously found solution. The average number is approximately about 15. Best results are obtained within range of <0.3; 0.5>. Time spend for each value of $\lambda$ is approximately same as seen on fig. 5. Each iteration mean one computing with different value of $\lambda$. For each value of $\lambda$ we made 10 iterations. Best found solution (fig. 6) is best for range of $\lambda$ within <0.3; 0.5>.
We made same tests on another function that is called Griewangk’s function (eq. 8) with similar results as with Sphere model. We obtain best results for range of $\lambda$ within $<0.4; 0.8>$ for time consuming (fig. 7). The best solution is at the point of zero (fig. 8) and modification of algorithm did not have so strong effect on results.
4 Conclusion

We introduced the concept of modification of simulated annealing for parallel computing of local search procedures to help understand what effect the created algorithm is having on the search in a defined space. By doing this, we can better evaluate if an algorithm works well as we expect, or if something different is happening. This helps to remove the ad hoc trial and error testing of meta-heuristics, which has become common in the literature. By doing it this way, we have also gained some understanding in how this algorithm works for each problem type we tested.

We have shown some basic heuristic of simulated annealing algorithm used in local search procedures that can help us understand how this is working. We have shown some methods based on random, population, local searching, and weights.

References:


Acknowledgment:

This paper was supported by grant P403/10/P334 of Czech science foundation.