Production Systems and Information Engineering
Volume 10 (3), pp. 129–136
doi: 10.32968/psaie.2022.3.10

COMPARISON OF COOLING STRATEGIES IN SIMULATED ANNEALING ALGORITHMS FOR FLOW-SHOP SCHEDULING

József Miliczki
University of Miskolc, Hungary
Institute of Information Technology
miliczki.jozsef@student.uni-miskolc.hu

Levente Fazekas
University of Miskolc, Hungary
Institute of Information Technology
levente.fazekas@uni-miskolc.hu

Abstract. Flow-shop scheduling is considered, where the order of operations must be the same for each job to minimize the maximum completion time. The Simulated Annealing algorithm is a standard approximate solution method in scheduling and optimization in general. Since the algorithm depends on cooling as a heuristic to generate better approximations, choosing the strategy with which the temperature decreases can affect the final result.

Keywords: flow-shop, scheduling, heuristics, simulated annealing, annealing strategies

1. Introduction

Flow-shop scheduling is considered, where the order of operations must be the same for each job to minimize the maximum completion time. Simulated Annealing algorithm is based on a local search algorithm with a stochastic criterion for accepting worse solutions. This algorithm is based on the Metropolis algorithm for simulating physics systems subject to a heat source [1]. Since the algorithm depends on cooling as a heuristic to generate better approximations, choosing the strategy with which the temperature decreases can affect the final result.

The $\alpha|\beta|\gamma$ formal classification scheme was introduced by Graham et al., where the resource environment, the characteristics of jobs, and the objective function are described by α, β, γ, respectively. The permutation flow-shop

$(F_m|perm|C_{max})$ scheduling problem is a classic example of production scheduling first formally introduced by Johnson et al. [2]. It is a one-way, multi-operation, shop-level problem where the jobs cannot precede one another, and the objective is to minimize the total makespan.

The flow-shop problem is formulated as N_J number of jobs processed on N_R number of resources of machines in a strict order. Each job consists of the same number of operations as the number of machines. The processing times of these operations are denoted as $p_{i,j}, i \in 1, 2, \ldots, N_J, j \in 1, 2, \ldots, N_R$ where i is the job, and j is the machine index. Only one operation can take place on a machine, and only one machine can operate on a job at a given time. We want to find a sequence of jobs s, such that the time required to complete all operations is minimized ($C_{max} \to \min$).

The flow-shop problem is proven to be NP hard [3, 4]. The best course of action is to use a meta-heuristic algorithm to come as close to a possible optimal solution without any certainty [5, 6]. The most common examples of these meta-heuristic algorithms are Variable Neighborhood Search [7], Simulated Annealing [8], and Genetic Algorithms [9, 10]. In this paper, we consider and compare different cooling strategies for the Simulated Annealing algorithm introduced by Kirpatrick et al. [8]. To benchmark each strategy, we used a benchmark set proposed by Taillard [11].

2. The Simulated Annealing Algorithm

The main premise of using Simulated Annealing is to use a cooling strategy to calculate the probability of accepting a worse sequence than what was currently the base for the neighbourhood searches. The probability of acceptance P_t can be calculated as such:

$$\Delta = f^* - f(s_t), \tag{2.1}$$

$$P_t = \begin{cases} e^{\frac{-\Delta}{T_t}}, & \text{if } \Delta \geq 0 \\ 1, & \text{if } \Delta < 0 \end{cases} \tag{2.2}$$

where f is the object function which is to be optimized, t is the iteration count, s_t is the current solution, f^* is the best solution found until iteration t, $f(s_t)$ is the value of the object function for the current solution, and T_t is a strictly decreasing sequence with

$$\lim_{t\to\infty} T_t = 0. \tag{2.3}$$

In algorithm 1 we define the Simulated Annealing meta-heuristic search algorithm, where

- s_0 is the initial solution,
- s is the base of the neighborhood,
- s_{best} is the best solution found so far,
- t_{max} is the maximum iteration count,
- t is the current iteration,
- T_t is the temperature in iteration t,
- s_n is the best candidate in a given neighborhood,
- n_c is the number of neighbors per neighborhood,
- s_{new} is neighbor of s,
- f is the object function,
- f^* is the best object function value found until iteration t.

Algorithm 1 Simulated Annealing

```
$s \leftarrow s_0$
$s_{best} \leftarrow s$
**for** $t \leftarrow 0$ to $t_{max}$ **do**
    $T_t \leftarrow$ Temperature($t$)        ▷ Computing temperature based on chosen annealing strategy
    $s_n \leftarrow s_0$
    **for** 1 to $n_c$ **do**
        $s_{new} \leftarrow$ neighbour($s$)
        **if** $f(s_{new}) < f(s_n)$ **then**
            $s_n \leftarrow s_{new}$
        **end if**
    **end for**
    **if** $P_t(f^*, f(s_n), T_t) \geq rand()$ **then**
        $s \leftarrow s_n$        ▷ If the condition (probability) for accepting a worse sequence is met or the found solution is superior, we enact $s_n$ as the new base
    **end if**
    **if** $f(s) < f^*$ **then**
        $s_{best} \leftarrow s$
    **end if**
**end for**
**return** $s$
```

3. Cooling Strategies

A practical implementation of the Simulated Annealing algorithm requires the generation of finite sequences decreasing values of temperatures T and a finite number of state transitions for each temperature value by using a cooling schedule.

The following three parameters are present in the cooling schedule introduced by Kirpatrick et al. [8]:

First, T_0 is the initial value of the temperature. It must be high enough that any new solution generated in a transition state should be accepted with a probability of 1.

The temperature decrease function is generally an exponentially decreasing function $T_t = t_0 - \alpha k$, where α is a constant smaller than 1.

The number of state transitions L for each temperature value.

The choice of cooling strategy will significantly affect the performance of Simulated Annealing so that the right approach can make a vast difference. We can identify multiple approaches when calculating the temperature based on time elapsed and a starting temperature as parameters. Choosing the right starting temperature T_0 is just as important as setting the lower boundary. With a proper setup of these two bounds, each strategy can be finely tuned to meet specific needs.

3.0.1. Additive Cooling Strategies

In *Additive* cooling, we must consider two additional parameters: the number of cooling cycles n, and the final temperature of the system T_n, a term that decreases with respect to the elapsed time t. Four variants are considered, based on the formulae proposed by B. T. Luke in 2005 [12]:

Linear Additive cooling (equation 3.1) is achieved by adding a term to the final temperature T_n that decreases linearly with respect to the elapsed time t:

$$T(t) = T_n + (T_0 - T_n)\left(\frac{n-t}{n}\right) \tag{3.1}$$

In *Exponential Additive* cooling (equation 3.2), the temperature decrease is computed by adding a term to the final temperature T_n that decreases in inverse proportion to an exponential function based on the elapsed time t:

$$T(t) = T_n + (T_0 - T_n)\left(\frac{n-t}{1+e^{\frac{2\ln(T_0-T_n)}{n}\left(t-\frac{1}{2}n\right)}}\right) \tag{3.2}$$

Trigonometric Additive cooling (equation 3.3) computes the temperature by adding a term to the final temperature T_n that decreases in proportion to the cosine of the elapsed time t:

$$T(t) = T_n + \frac{1}{2}(T_0 - T_n)\left(1 + \cos\frac{t\pi}{n}\right) \tag{3.3}$$

Quadratic Additive cooling (3.4.) decreases the temperature by adding a term to the final temperature T_n that decreases in proportion to the square of the elapsed time t:

$$T(t) = T_n + (T_0 - T_n)\left(\frac{n-t}{n}\right)^2 \tag{3.4}$$

3.0.2. Multiplicative Cooling Strategies

In multiplicative cooling, the system temperature T at time t is computed by multiplying the initial temperature T_0 by a factor that decreases with respect to the current time t. Four variants are considered:

Linear Multiplicative cooling (equation 3.5) decreases the temperature by multiplying the initial temperature T_0 by a factor α that decreases in an inverse proportion to the elapsed time t.

$$T(t) = \frac{T_0}{1+\alpha t} \tag{3.5}$$

In *Quadratic Multiplicative* cooling (equation 3.6), the temperature is decreased by multiplying the initial temperature T_0 by a factor α that decreases in an inverse proportion to the square of the elapsed time t:

$$T(t) = \frac{T_0}{1+\alpha t^2} \tag{3.6}$$

Exponential Multiplicative cooling (equation 3.7), as proposed by Kirpatrick et al. [8] is used as the reference for comparison among different cooling criteria. The temperature is obtained by multiplying the initial temperature T_0 by a factor α that decreases exponentially with respect to the elapsed time t:

$$T(t) = T_0 \cdot a^t. \tag{3.7}$$

Logarithmic Multiplicative cooling (equation 3.8) is based on the asymptotical convergence condition, but it incorporates a factor α of cooling speedup, thus making it possible to use in practice. The temperature is decreased by multiplying the initial temperature T_0 by a factor α that decreases in an inverse proportion to the natural logarithm of the elapsed time t:

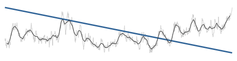
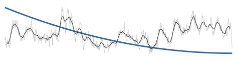
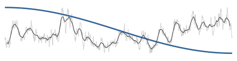
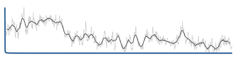
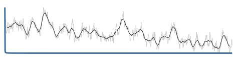
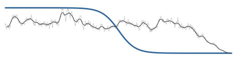
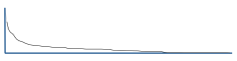
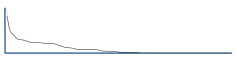
$$T(t) = \frac{T_0}{1 + \alpha \log(1 + t)} \tag{3.8}$$

4. Comparison of Cooling Strategies

In this chapter, we compare the performances achieved by the introduced cooling mechanisms using the Taillard benchmark set[11]. Every test has been conducted using the same random number seed and production times described in the first test of the 100 jobs, 20 machines set with an upper bound of 5770. The performed number of iterations is 1000, each considering a neighbourhood of 20. The initial temperature is 10^6, with α variable set to 0.8.

Table 1. contains the algorithms and their parameters, the final makespan, the final temperature reached, as well as the changes in temperature and makespan throughout the runtime of the algorithm.

Table 1. Benchmark results

Strategy	T_0	$T_{t_{max}}$	α	C_{max}	
Linear Additive	10^6	1000	–	7459	
Quadratic Additive	10^6	3.20	–	7459	
Trigonometric Additive	10^6	4.53	–	7459	
Linear Multiplicative	10^6	62.56	0.8	7227	
Logarithmic Multiplicative	10^6	50.05	0.8	7199	
Exponential Additive	10^6	3.33	–	6851	
Exponential Multiplicative	10^6	0	0.8	6588	
Quadratic Multiplicative	10^6	0	0.8	6536	

From table 1 it is evident that choosing the proper cooling schedule alongside the starting temperature has a significant impact on the search results when it comes to the Simulated Annealing Algorithm. One important thing to note is the remaining temperature at the end of the search $T_{t_{max}}$. Where the remaining temperature is higher, the search behaves more like a random search algorithm throughout all the iterations, resulting in a higher C_{max} value. The Logarithmic multiplicative schedule is the only schedule mentioned here that will never reach a temperature of 0, rendering it impractical for real-world use.

5. Conclusion

In addition to the findings and use cases presented in this paper, we also found that these cooling schedules can be interpreted differently for other scheduling tasks. The advantage of these cooling schedules is the ability to fine-tune the approach taken to scheduling when it comes to local search algorithms. These techniques can prove helpful when tackling more and more complex problems.

The results are encouraging to continue our research, incorporating other methods, parameters, models and techniques into local search algorithms, applying them to real-world applications.

References

[1] Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E.: Equation of state calculations by fast computing machines. *The journal of chemical physics*, **21**(6), (1953), 1087–1092, URL https://doi.org/10.1063/1.1699114.

[2] Johnson, S. M.: Optimal two-and three-stage production schedules with setup times included. *Naval research logistics quarterly*, **1**(1), (1954), 61–68, URL https://doi.org/10.1002/nav.3800010110.

[3] Lageweg, B., Lawler, E. L., Lenstra, J. K., and Rinnooy Kan, A.: Computer aided complexity classification of deterministic scheduling problems. *Stichting Mathematisch Centrum. Mathematische Besliskunde*, **1**(BW 138/81).

[4] Lageweg, B., Lenstra, J. K., Lawler, E., and Kan, A. R.: Computer-aided complexity classification of combinational problems. *Communications of the ACM*, **25**(11), (1982), 817–822, URL https://doi.org/10.1145/358690.363066.

[5] Knust, S.: Complexity results for scheduling problems. http://www2.informatik.uni-osnabrueck.de/knust/class/ [2022-06-09], 2009.

[6] Leung, J. Y.: *Handbook of scheduling: algorithms, models, and performance analysis*. CRC press, 2004.

[7] Mladenović, N. and Hansen, P.: Variable neighborhood search. *Computers & operations research*, **24**(11), (1997), 1097–1100, URL https://doi.org/10.1016/S0305-0548(97)00031-2.

[8] Kirkpatrick, S., Gelatt Jr, C. D., and Vecchi, M. P.: Optimization by simulated annealing. *science*, **220**(4598), (1983), 671–680, URL https://doi.org/10.1126/science.220.4598.671.

[9] Fraser, A. S.: Simulation of genetic systems by automatic digital computers i. introduction. *Australian journal of biological sciences*, **10**(4), (1957), 484–491, URL https://doi.org/10.1071/BI9570484.

[10] Fraser, A., Burnell, D., et al.: Computer models in genetics. *Computer models in genetics.*

[11] Taillard, E.: Benchmarks for basic scheduling problems. *european journal of operational research*, **64**(2), (1993), 278–285, URL https://doi.org/10.1016/0377-2217(93)90182-M.

[12] Luke, B. T.: Simulated annealing. *Retrieved June*, **8**, (2005), 2005.