American Journal of Applied Sciences

Original Research Paper

A New Innovative Cooling Law for Simulated Annealing Algorithms

Antonella Certa, Toni Lupo and Gianfranco Passannanti

Dipartimento di Ingegneria Chimica, Gestionale, Informatica, Meccanica (*DICGIM*),
Università Degli Studi di Palermo, Viale Delle Scienze, 90128, Palermo, Italy

Article history
Received: 13-04-2015
Revised: 27-06-2015
Accepted: 04-07-2015

Corresponding Author:
Toni Lupo
Dipartimento di Ingegneria Chimica, Gestionale, Informatica, Meccanica (DICGIM), Università Degli Studi di Palermo, Viale Delle Scienze, 90128, Palermo, Italy
Email: toni.lupo@unipa.it

Abstract: The present paper proposes an original and innovative cooling law in the field of Simulated Annealing (SA) algorithms. Particularly, such a law is based on the evolution of different initial seeds on which the algorithm works in parallel. The efficiency control of the new proposal, executed on problems of different kind, shows that the convergence quickness by using such a new cooling law is considerably greater than that obtained by traditional laws. Furthermore, it is shown that the effectiveness of the SA algorithm arising from the proposed cooling law is independent of the problem type. This last feature reduces the number of parameters to be initially fixed, so simplifying the preliminary calibration process necessary to optimize the algorithm efficiency.

Keywords: Simulated Annealing, Cooling Law, Job-Shop, Project Crashing

Introduction

Complex nonlinear optimization problems require specific resolution techniques. These problems are often characterized by a solution space that presents many local optima (for the sake of simplicity it will be made reference to a cost function minimization). In these cases, local search algorithms, as the classical descent neighborhood search method, have a heavy drawback: The optimization algorithm generally converges towards a local minimum.

To avoid getting trapped in a local minimum, the optimization algorithm must allow to accept worse solutions than the current one. Several kinds of algorithms have been developed for this purpose and they differ for the acceptance criteria of a pejorative solution. Among such algorithms it is possible to remember the Taboo Search (TS) and the Simulated Annealing (SA). Another class of algorithms, i.e., the Genetic Algorithms (GAs), is based on the evolution of a set (population) of initial solutions. Such classes of algorithms have been applied to several different fields of research. Our interest is addressed to the SA algorithms, in particular to the cooling law, which is fundamental for the SA efficiency.

The reminder of the paper is organized as follows: The most recent proposed cooling laws for SA are presented first, followed by a detailed description of the new proposed cooling law; the effectiveness and efficiency of the new proposal are evaluated next. Meanwhile, several testing problems of the proposed law are presented in detail. Hence, conclusion of this study is provided as well.

Simulated Annealing and Cooling Schedule

The logic of a SA algorithm is well known by the literature. We briefly recall its steps, drawn by (Kirkpatrick *et al*., 1983) and inspired to the previous studies of (Metropolis *et al*., 1953). An initial feasible solution, called seed, is perturbed and replaced by the new one if better. If the new solution is worse, then it is accepted with a probability calculated by the well-known law: $Prob = exp\{-\Delta F/T\}$, where ΔF is the cost increment. The procedure is iterated on the current seed and at the same time the "temperature" T is reduced. This progressive reduction makes less probable the acceptance of a more expensive solution, until this probability is practically reduced to zero. In the last case, a state called "frozen" is reached and the algorithm is stopped. From a theoretical point of view, an opportune choice of the SA parameters and functions can lead to the optimal solution independently of the initial seed. In particular, one refers to the initial temperature, T_o and the cooling law that, as shown by some researchers (Geman and Geman, 1984; Hajek, 1988), should assume the form:

$$T_r = C / ln(1+r) \quad (1)$$

Being r the progressive number of the analyzed solutions and C an opportune constant depending on the maximum cost difference between a random solution and its neighborhood. Nevertheless, apart from the difficulty to determine the value of C, the achievement of the frozen state could require unacceptable run-times. Some proposals of modification of the cooling law attempt to compensate for such a drawback. For example, Lundy and Mees (1986) propose the following cooling law:

$$T_r = T_{r-1} / \left(1 + \beta T_{r-1}\right) \tag{2}$$

However, the simplification does not assure the achievement of the optimal solution. In order to further on simplify, the temperature is not usually reduced at each new solution, namely the algorithm is articulated in cycles characterized by a constant temperature and a given number of solutions, *nrep*, is analyzed for each cycle. Within the generic c^{th} cycle, the probability of transaction from a state to another one (i.e., the acceptance of a new solution from the current one) is constant and it only depends on the two involved states. Therefore, it is a homogenous Markov's process and the value of *nrep* should be such that steady conditions are reached before temperature reduction. This reduction implies a contemporary reduction of the transition probability and thus the achievement of stationary conditions is always slower. Then the *nrep* value should be progressively increased. Even if constant temperature cycles are employed, a cooling law is however necessary and it should determine a trend of temperature similar to the slow downgrade arising from the relation (1).

Interesting proposals have been suggested by (Huang *et al*., 1986):

$$T_c = T_{c-1}\, exp\left(-\beta T_{c-1} / \sigma_{c-1}\right) \tag{3}$$

and by (Aarts and Korst, 1989):

$$T_c = T_{c-1} / \left[1 + T_{c-1}\, ln\left(1+\delta\right) / \left(3\sigma_{c-1}\right)\right] \tag{4}$$

These relations also take into account the evolution of the algorithm by means of the term σ_{c-1}, that is the standard deviation of the cost values at the $(c-1)^{th}$ Markov chain. Both these cooling laws could be connected with the concept of the specific heat introduced by (Kirkpatrick *et al*., 1983). The specific heat is related to the costs variance at a given temperature. A high variance is index of high distance from the convergence conditions and thus the temperature can be more rapidly reduced. However, a simple geometric law is usually preferred:

$$T_c = \alpha T_{c-1} \tag{5}$$

with $0< \alpha <1$.

About the choice of the parameters involved in Equation (5), exact rules do not exist, but just sensible indications. So, the initial value of temperature, T_o, is chosen in order to accept a strongly pejorative solution with a high probability that allows, at least at the beginning, "to wander" in the overall solutions space and thus the final solution results to be independent of the initial seed. Usually, in a preliminary tuning phase, the value of T_o is determined so that the fraction of accepted pejorative solutions is very high (for example 0.9).

A very gradual reduction of the temperature requires an α value very close to the unity. Used values usually belong to the range [0.8, 0.99]. For increasing α values, the achievement of the frozen state requires an increasing number of cycles and consequently an increasing run time. Obviously, the latter also depends on the value of *nrep* and thus it is fixed with relation to the other parameters and consistently with the available elaboration time.

A different interesting temperature control scheme is proposed by (Azizi and Zolfaghari, 2004). It is a particular cooling law (or better, a heating law) that dynamically modifies the temperature on the base of the search path. The control function of the temperature is the following:

$$T_c = T_o + \lambda ln\left(1+r\right) \tag{6}$$

where, T_o is the minimum value that the temperature can assume, λ is a coefficient that determines its increase rate and r the number of consecutive solutions having a greater cost than the current one. When an improvement is obtained, r is set equal to zero.

Dowsland (1993) takes contemporaneously into account two different functions for the temperature control: Besides a cooling law, a heating law is considered to gradually increase the temperature, if necessary.

To our knowledge, the most recent literature has not proposed new cooling laws. The literature about the SA is wide, but it deals with some specific applications (Kia *et al*., 2012; Leung *et al*., 2012), as well for multi-objective problems (Lin and Ying, 2013), or proposals of utilization of SA matched with other heuristics, as the Ant Colony (AC) algorithms (Sitarz, 2009), the TS or, more often, the GAs (Rong-Ceng, 2006; Zahrani *et al*., 2008).

The New Cooling Law

Our proposal arises from the idea of the specific heat (Kirkpatrick *et al*., 1983). For some kind of problems (for example unimodal problems), a little

value of σ, standard deviation of costs at a given temperature, can correctly induce to suppose to be close to the minimum value of cost. The closer the neighborhood of a solution to the same solution is, the truer this supposition could be. In such cases, the laws (3) and (4) find full justification. For other classes of problems, for example for combinatorial optimization problems, a high or a little value of σ could not be meaningful of the particular reached conditions. For example, let consider the function represented in Fig. 1 (Michalewicz, 1999).

The more the solution approaches the minimum (or the maximum) of the function, the more its neighborhood is characterized by a high variability. On the contrary, if the neighborhood is very extended, the variability is nearby constant, whatever the reached solution is. In our opinion, the cost variability among the solutions is very important, even if the variability to be taken into account is not that related to the solutions visited at a given temperature.

Specifically, once a final solution of the problem has been obtained, a method to look for obtaining a better solution consists of repeating the overall procedure by beginning from a different seed, if the available time permits that. The repetition of the procedure, eventually simplified by acting on α, *nrep*, T_o and on the final temperature, can present more benefits than an unique prolonged procedure.

Let the procedure can be iterated *Nseed* times. Instead of repeated runs, the iterations can be simultaneously carried out. For example, at the c^{th} step, $nrep(c)$ moves could be executed on each one of the *Nseed* solutions that constitute the evolution of as many original random seeds. The analysis of the contemporary evolution from the original seeds could supply information that might be lost if each seed is singly developed till the end. In particular, at each step, *Nseed* solutions are available, arising from different original seeds characterized by very different costs. They are converging to solutions that, even if not optimal, could be near to the optimal one. Hence one can rightly think that their costs are mutually approaching. The mutual "distance" among the *Nseed* solutions at the end of the c^{th} step can be used as indirect measure of how much the optimal solution is still faraway to be reached. On the other hand, when a solution is far from the optimal one, the probability to accept worse solutions should be high whereas it should decrease as much as nearer it is to the optimal one. Thus the acceptance probability can be linked to the distance among the current solutions and this distance could be measured by the standard deviation, η, related to the *Nseed* cost values. After all, the use of the following cooling law is here proposed:

$$T_c = k\eta_{c-1} \tag{7}$$

where, η_o is the standard deviation of the costs of the original random solutions and k is a constant opportunely chosen as explained in the next section. It is possible to think of more complex relations between temperature and η but the optimal results obtained by the proposed law do not justify their use.

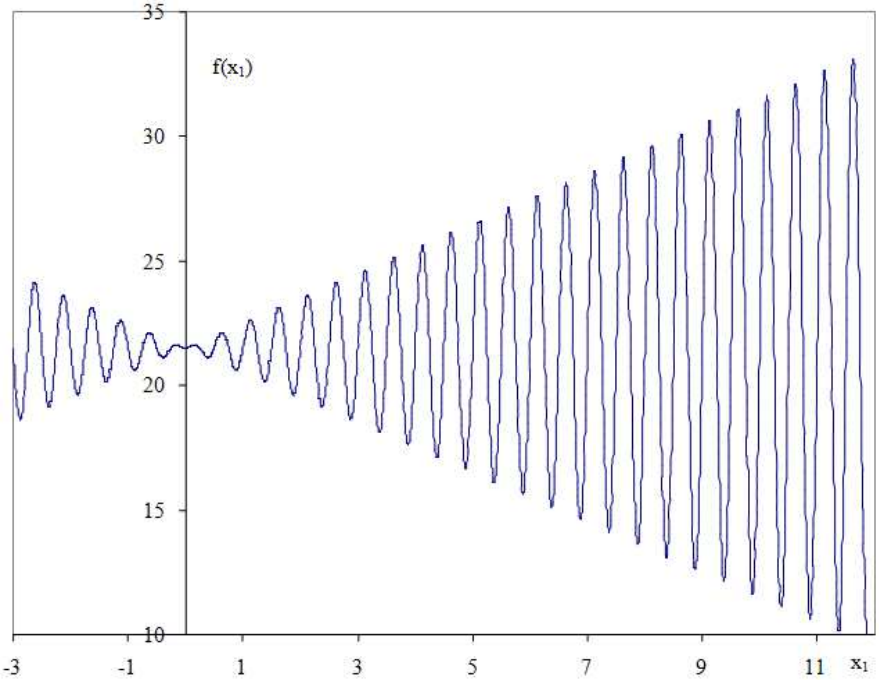

Fig. 1. Drawing of the function $f(x_1, x_2) = 21.5 + x_1 \sin(4\pi x_1) + x_2 \sin(20\pi x_2)$ with $x_2 = 0$

Note that both η and σ are the costs standards deviations. A different symbol has been used since the value of σ in Equation (3) and (4) is calculated on costs related to the solutions visited during a cycle, whereas η is the standard deviation of the current *Nseed* solutions at the end of a cycle.

About the Values to Assign to the Parameters

Implementing a SA with the new cooling law requires to fix some parameters, *Nseed*, *nrep*, *k* and a closing criterion. Generally, the selection of the values to be assigned to the parameters of a heuristic algorithm is not based on rigid laws. In the following, general suggestions are just proposed. The authors hope to have been guided by the common sense.

A closing criterion, whatever it may be, can not disregard the time available for the solution search. On the other hand, this time allows at analyzing a limited number of solutions. Then, the closing criterion can explicitly refer to the global number of solutions, S_{tot}, that one wants to analyze.

As for the population of a GA, the value of *Nseed* could be binded to the solutions space dimension to have information about different zones. However, this fact is not important for an SA, since, as it has been shown, it is able to be free from the initial solution and the final result should be independent of it. Moreover, since S_{tot} must be divided among the original seeds, the higher the value of *Nseed* is, the less extensive the search on each seed can be. After all, a relation of the following kind is proposed as a good compromise among the various exigencies shown:

$$Nseed \cong S_{tot}^{0,2} \tag{8}$$

As to *nrep*, this parameter is not very critical. An its low value imposes a frequent, but probably superfluous, up-to-dating of η. Its value could be fixed about some hundred: The analysis of *nrep*·*Nseed* solutions could show a more or less significant change of the standard deviation.

The most critical parameter certainly is *k* and it isn't possible to make hypotheses about its right value. All researchers emphasize the importance of a careful tuning of the parameters. Such a tuning phase often is laborious and time consuming, while this time could be profitably spent for a longer run of the algorithm. Well, the proposed algorithm does not require such a set-up phase. Really, authors are positively persuaded that *k* is a constant independent of the kind of the specific faced problem. At the moment, they could not propose a proof for this assertion and so it is just a conjecture. A formal proof could be the object of a subsequent research during which a better value of *k* than that one used in this study could be determined.

Effectiveness and Efficiency of the New Proposal

The Authors do not propose a new algorithm for the resolution of a specific problem, but a change, of general validity, on the traditional cooling law used within the SA algorithms. Therefore, the verification of the effectiveness of the proposal should be made with relation to algorithms of the identical type, in terms of formulation except for the cooling rule. Hence, one should consider an algorithm already researched in literature and it should be implemented in two versions, namely with and without the change. Unfortunately, the literature proposals on SA (not existing for the problem of crashing) do not specify the values of the different parameters to be chosen, but intervals. Their setting depends on the user that has to determine their best combination with relation to the specific data of the faced problem. Therefore, we had to search for the optimal parameters in order to successively state that our proposal is much better. As it is obvious, the procedure would have been widely questionable. We believed to operate in a different way.

Specifically, we developed a SA algorithm for a given job-shop scheduling problem and we implemented it twice, namely one with the traditional cooling rule and another one with the new proposal, unchanging the other parameters. Authors believe improbable that an unreasonable change within an ineffective algorithm can considerably improve its performance. Nevertheless, ordering to prevent a possible and rightful criticism in this sense, Authors tested the algorithm in its "classical" formulation by comparing its results with the ones obtained by means of Monte Carlo simulation.

One can state that the new proposal is valid just because of the dealt problem (scheduling with sequence dependent setup times). In order to prevent such a criticism, the whole procedure has been repeated even for a class of problems with different features (project crashing).

Therefore, the verifications have been articulated in more steps:

- Development of two "classic" algorithms, named by the acronyms of the two problems: Sequence Dependent Setup (SDS) and Project Crashing (PC)
- Monte Carlo simulations with the same kind of coding and decoding employed in the previous algorithms
- Implementation of SDS and PC and comparison with the Monte Carlo simulation results
- Implementation of the new cooling law for the two typologies of problems (New_SDS and New_PC) and comparison with the previous results

Testing Problems

As just said, the proposal advanced in the present paper has been tested on two different classes of problems: (a) the

job shop scheduling with sequences-dependent set-up times and (b) the project crashing problem.

A brief presentation of the two kinds of problem is given.

Job Shop Scheduling with Sequences-Dependent Set-Up Times, SDS

The SDS is a typical industrial problem. The set-up consists in those operations, as fixturing and tooling, that are needed to prepare a workstation. The optimization usually aims at determining the schedule that implies the minimum completion time (makespan). The problem is NP-hard (Pinedo, 2002). The interest of the industry for the problem and the challenge represented by its difficulty, have led many researchers to face such class of problems for various typologies of production systems (Low, 2005; Jin *et al*., 2009). In the following we will pay our attention to the job-shop scheduling problem. In literature, this field has not been very widely researched. Apart from some proposals that usually utilize the mixed integer programming for specific production systems (Luh *et al*., 1998; Mason *et al*., 2005), problems of big dimensions are usually solved by heuristics. Therefore, resolution proposals with TS (Hertz and Widmer, 1996), with dispatching rules (Kim and Bobrowski, 1997), with GA (Sun *et al*., 2003; Cheng *et al.* 1996), with AC (Timur *et al*., 2012), with integration of GA and dispatching rules (Cheung and Zhou, 2002) can be found in the literature. Lin *et al*. (2009) propose an application of SA for the problem of our interest but in the flow-shop field. A wide bibliography about scheduling with SDS is given in the paper of Zhu and Wilhelm (2006).

The Project Crashing Problem

In the project management field, the PC concerns the determination of the length of each activity, so that the project completion time L does not overcome a maximum value, UpL, with the aim of minimizing the sum of the direct costs related to the activities, C. No hypothesis on the resource availability is introduced. The problem is then formulated as follows:

$$\min C \tag{9}$$

Subject to:

$$L \le UpL \tag{10}$$

The crashing technique for the dual of the previous problem can be described as a specific type of project schedule compression technique. The latter is performed to decrease the total project schedule length after analyzing a number of alternatives to determine how to get the maximum schedule duration compression for the least additional cost (PMI, 2013). In the last case, typical approaches for crashing a schedule include reducing activity durations and increasing the assignment of resources.

As to the approaches proposed to solve such a kind of problems, GAs are widely employed for linear (Li and Love, 1997; Leu and Yang, 1999), or quadratic (Li *et al*., 1999) or discrete (Feng *et al*., 1997) time-cost relations.

Among the more recent approaches it is possible to cite (Tung, 2007), who develops and tests a particle swarm optimization algorithm, Aghaie and Mokhtari (2009) whose approach is based on the AC optimization metaheuristic and Monte Carlo simulation technique, Liberatore and Pollack-Johnson (2006) who propose a quadratic mixed integer programming approach for reducing the project completion time.

The SDS Algorithm

A brief description of the algorithm is given in Appendix A1.

The SDS has been applied to the SWV01 problem, 10 machines and 20 jobs, proposed by (Storer *et al*., 1992). The setup times, absent in SWV01, have been simulated by an integer uniform distribution U[1;20]. Due to the utilized coding, the dimension of the solution space is greater than 10^{191}. The Monte Carlo simulation stretches up to $N_1 = 10^7$ solutions. The results are reported in Fig. 2.

Note that the minimum value obtained is equal to 2056. An unbiased non parametric estimate of the probability of obtaining values lower than 2056 is given by the ratio $1/(N_1+1)$ and then this probability takes the value $P_1=10^{-7}$. Moreover, the standard deviation of the frequency distribution is equal to s = 85.7.

By assuming that $3s$ is a good approximation of the maximum difference between a random solution and its neighborhood, relation (1) allows to calculate $T_1 = 370$. Obviously the parameter s (or other useful parameters) can be evaluated with a less number of random solutions. For example, Xinchao (2010) generates ten random solutions and calculates the difference Δ between the worst and the best solution. Afterwards T_o is calculated by the relation $T_o = -\Delta/\ln P_o$, where P_o is equal to the selected initial acceptance rate for the worse solution.

As final condition, a probability $P_{fin} = 0.005$ has been fixed for the acceptance of a solution that is worse for more than a unit of makespan. From the last consideration arises a final temperature $T_{fin} \cong 0.4$. By assuming a geometric cooling law with $\alpha = 0.99$, a number of Markov chains equal to 680 is obtained. If *nrep* grows according to the relation $nrep_c = nrep_{c-1} / \alpha^{0.1}$ and one wants to analyze 10^6 solutions, then $nrep_1 = 1026$. Figure 3 reports the mean value of makespan over 10 independent runs (central line).

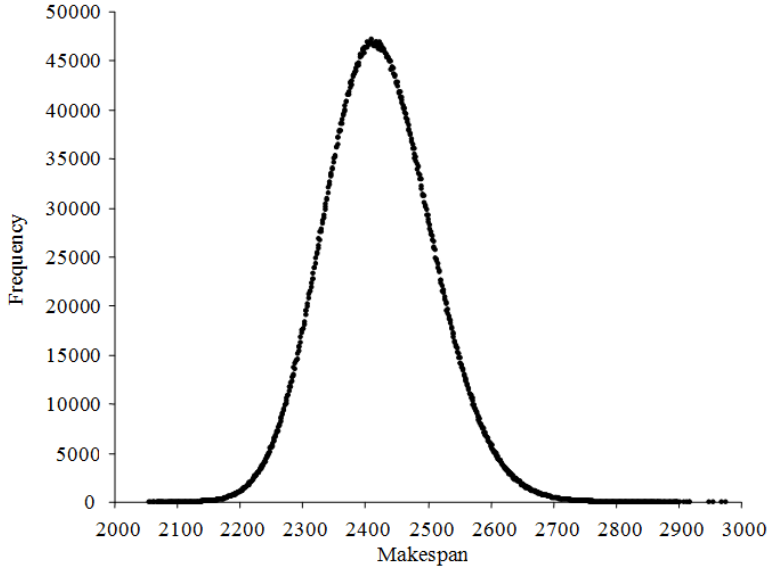

Fig. 2. Monte Carlo simulation for the SDS problem

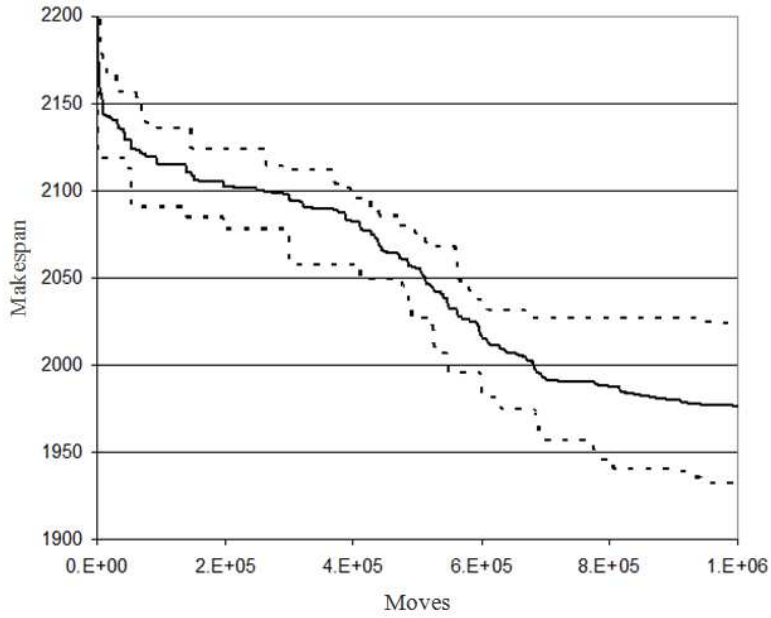

Fig. 3. Results for the classic SA algorithm

The two external lines define the range which includes the best values achieved in the ten simulations. The considerable reduction of makespan, in comparison with the best value obtained by the simulation, is evident. Although P_1 is very small and each run of the algorithm has only analyzed 10^6 solutions, all the final results are better than 2056, that is the best value obtained by analyzing 10^7 random solutions. From a probabilistic point of view, considering that the value of 2056 is reached after 450000 moves, the probability that it could casually occur within a run is around equal to 0.043 (already not significant at a level α = 5%), whereas the probability that it occurs in 10 successive runs is around 2.10^{-14}. Therefore, one can certainly affirm that the proposed algorithm is highly effective.

The PC Algorithm

A brief description of the algorithm is given in Appendix A2.

The precedence graph reported in the paper of (Arikan and Gungor, 2001) including 80 activities and 62 paths has been utilized (however, the related meaning was different from that attributed in the present work). The number of duration alternatives for each activity was

simulated from U[3;6]. In particular, 17 activities resulted to be characterized by 3 alternatives, 31 by 4, 16 by 5 and 16 by 6. Consequently, the dimension of the solution space is $2.56 \cdot 10^{50}$. The time lengths were simulated by means of the relations:

$$L(a,1) = U[20;40]\, L(a,j) = L(a,j-1) + U[10;20] \; with \; j = 2 \; to \; y(a)$$

and the costs by:

$$c(a,j) = 1000 / L(a,j) \; j = 1 \; to \; y(a)$$

The minimum duration of the project results to be L_{min} = 852 and the maximum one L_{max} = 2213. To the maximum duration corresponds a minimum cost C_{min} = 1049.

The objective to be achieved is that expressed by the relation (9) and (10). *UpL* is fixed by: $UpL(\tau) = L_{min} + \tau . (L_{max}-L_{min})$ and two different values of the parameter τ have been considered: $\tau_1 = 0.4$ and $\tau_2 = 0.8$.

The Monte Carlo simulation has been limited to $N_2 = 10^6$ solutions (the solutions space is considerably smaller than that one of the scheduling problem). The results are reported in Fig. 4.

Particularly:

$$C_{min}(\tau_1) = 1629 \; C_{mean}(\tau_1) = 1859.90 \; s(\tau_1) = 62.47$$

$$C_{min}(\tau_2) = 1401 \; C_{mean}(\tau_2) = 1731.15 \; s(\tau_2) = 77.41$$

For the implementation of the PC, initial and final values of the temperature have been determined by reasoning as for the scheduling problem, so obtaining:

$$T_1(\tau 1) = 270 \;\; T_1(\tau 2) = 335 \;\; T_{fin} = 0.4$$

Furthermore, by assuming $\alpha = 0.98$:

$$Markov's \; chains(\tau_1) = 323$$

$$Markov's \; chains(\tau_2) = 333$$

Assuming *nrep* as constant and limiting the analyzed solutions to 10^5, then:

$$nrep(\tau_1) = 310 \; and \; nrep(\tau_2) = 300$$

$$Moreover \; s_1 = s_2 = 6 \; were \; assumed$$

Ten independent runs have been executed and the results are reported in Fig. 5 and 6.

As for the SDS, the central line is referred to the medium cost. The two external lines indicate the interval within which the best values reached during the simulations are included. It is possible to affirm that the algorithm is very efficient: For both the cases (τ_1 and τ_2) the best value obtained by the simulation is reached just after the first cycles and the final costs are notably lower. The probability that such results casually happen is even lower than those of the SDS problem (<0.1% for a single run and $<10^{30}$ for the set of 10 runs) and then absolutely not significant. Furthermore, it is interesting to note that the final range of variability among the different solutions is very narrow and this suggests that the optimal solutions, even if not reached, are very near to the obtained results.

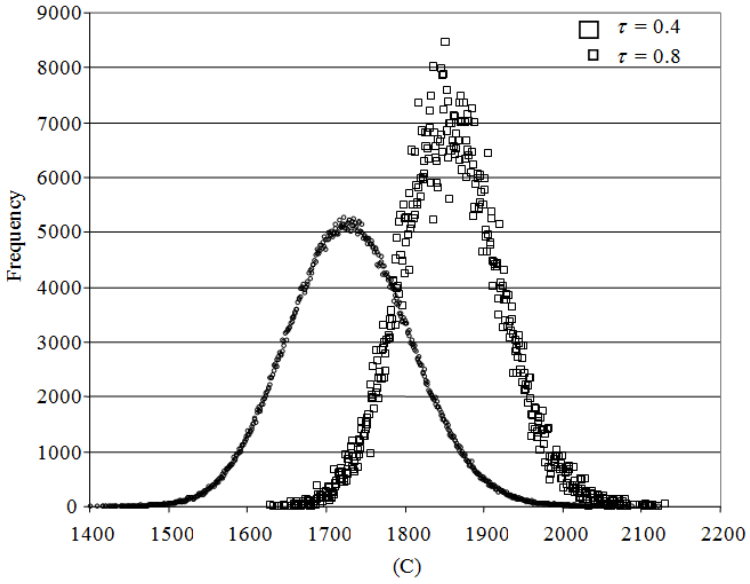

Fig. 4. Monte Carlo simulation for the PC problem

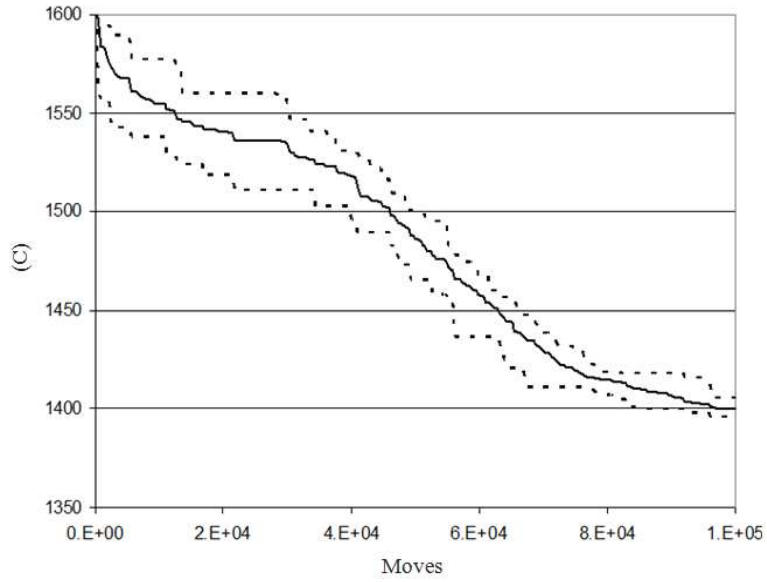

Fig. 5. Results for the classic SA algorithm ($\tau_1 = 0.4$)

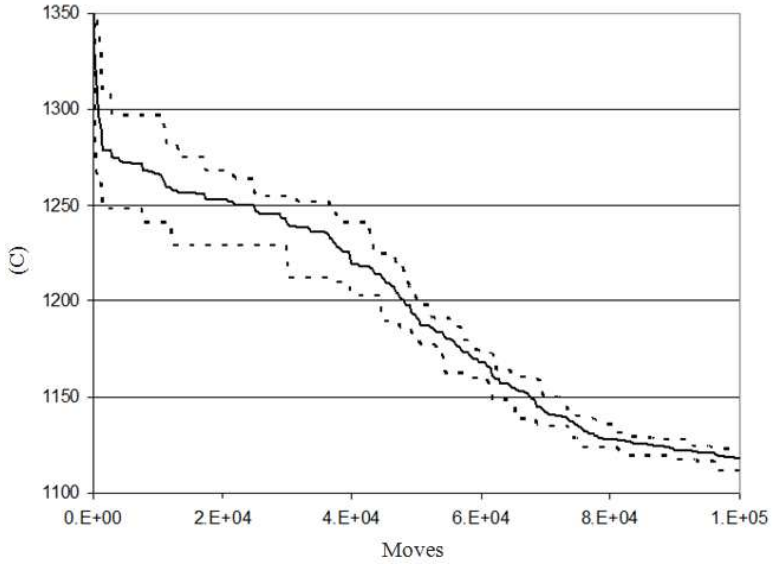

Fig. 6. Results for the classic SA cost algorithm ($\tau_2 = 0.8$)

The New_SDS Algorithm

In order to have a fair comparison with the SDS, the total number of analyzed solutions for each run test must be equal to the previously utilized value, that is 10^6. Following the suggestions earlier expressed, the value of *Nseed* has been determined making use of the relation (8), namely $Nseed = (S_{tot})^{0.2} = (10^6)^{0.2} \cong 16$. S_{tot} has been subdivided among the 16 seeds and partitioned in 150 cycles, each one including a constant number of moves *nrep* = 416.

About the value of *k*, the novelty of the approach does not allow to have indications coming from previous implementations. On the other hand, even if *k* is supposed to be independent of the considered problem, since its value can not be derived from previous theoretical considerations, it has to be determined by some experiments on the base of a specific problem. Then, some preliminary tests have been carried out to individuate the *k* value that supplies the best results. By these tests, referred to the SDS problem, the most suitable value for the parameter k seems to be 0.08. This value was used in the subsequent analyses.

Resuming, the values utilized to implement New_SDS have been:

$Nseed = 16;\ nrep = 416;\ number\ of\ cycles = 150;\ k = 0.08$

Figure 7 reports, as before, the mean value of the makespan over 10 independent runs (central line, a) and the interval in which all the best values are included. The analogous results (curves *a*', *b*', *c*') related to the SDS and already seen in Fig. 3 are reported for an easier comparison.

Note that:

- The mean makespan value obtained by the New_SDS (curve *a*) always results better than the best results obtained by the SDS (curve *b*')
- The worst values obtained by the New_SDS (curve *c*) result better than the mean value (curve *a*') related to the SDS

Actually, the comparison between two means needs to be performed by a statistical test (i.e., the t test) to verify that the difference is not just a random result. Considering that the two means position changes at the algorithm running, the test should be repeated at increasing the number of cycles, to also verify if differences exist in the convergence speed. In alternative, we preferred to compare 10 couples of profiles by means of a non-parametric test (i.e., the sign test, a very conservative criterion for the comparison of whole profiles). As it is possible to note in Fig. 7, the two sets of results are disjointed. As consequence, an empirical significance level of $(1/2)^{10}$ is obtained, significantly smaller than the classical $\alpha = 5\%$.

The fact that the results obtained by means of the New_SDS undoubtedly are better than those obtained by the SDS probably is the least significant result. What is surely remarkable is the fact that the final mean value of the SDS (1976.5) obtained after 10^6 trials is reached by the New_SDS after 80000 trials alone. Thus the most meaningful result is the greatest convergence speed of the new algorithm, that permits to reduce the run time of an order of magnitude at least.

The New_PC Algorithm

By reasoning as before made, the following parameters are assumed:

$$Nseed = (10^5)^2 = 10;$$
$$nrep = 100;\ number\ of\ cycles = 100;\ k = 0.08$$

Note that the value of *k* is just alike to that one utilized in the scheduling problem. Results in Fig. 8 and 9.

For the sake of simplicity the curves related to the mean values are not reported and the extreme conditions for the New_PC (curves *a* and *b*) and PC (curves *a*' and *b*') are only drawn. Even in such a case, comparing the two series of profiles, the sign test assures that differences between the series of results are highly significant and, as before, the most significant matter to be observed is the greatest convergence speed obtained by the new cooling law.

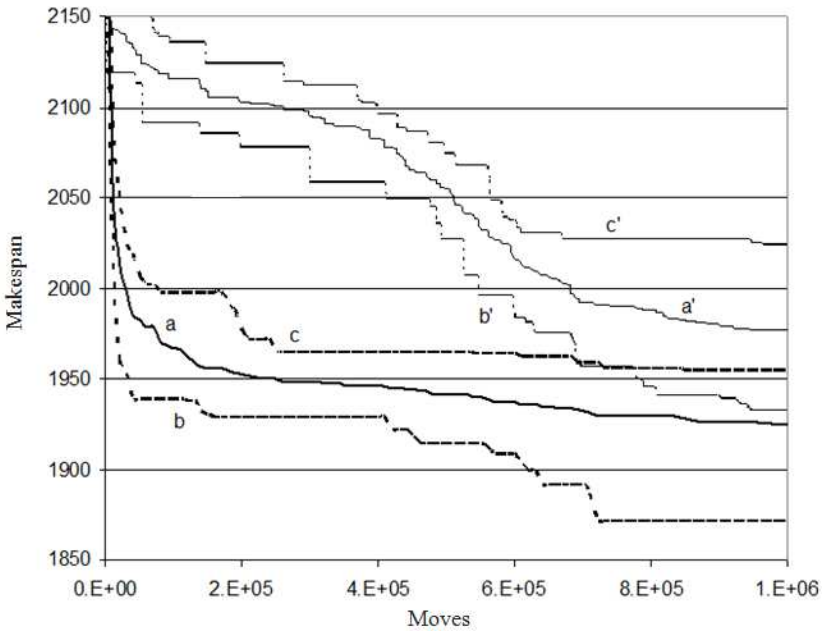

Fig. 7. Comparison between the new SA Vs the classic SA

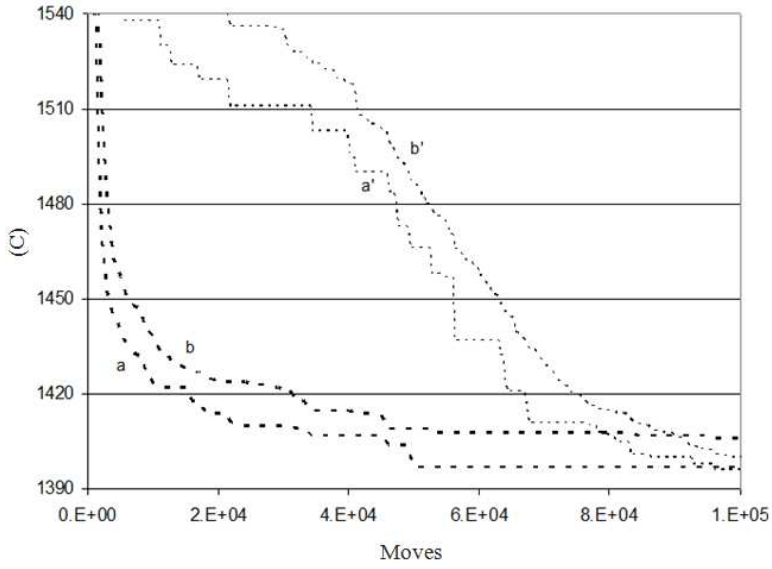

Fig. 8. Comparison between the new SA Vs the classic SA (τ = 0.4)

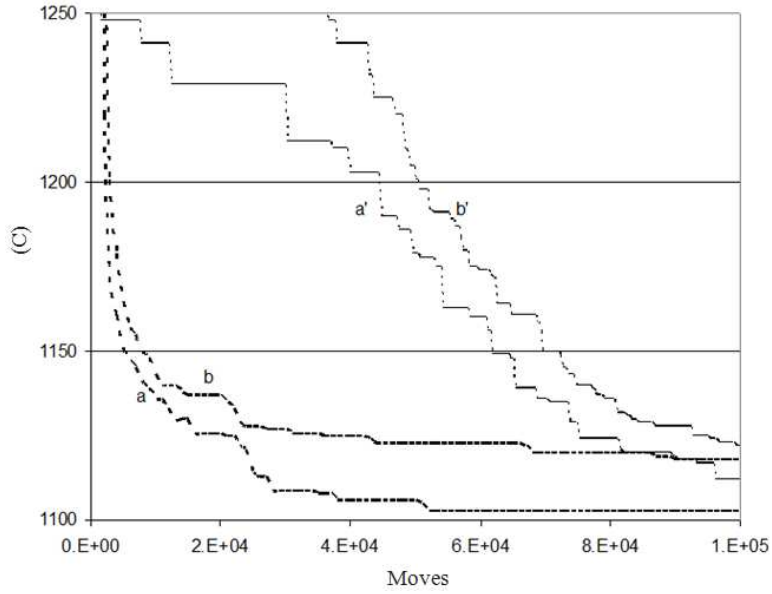

Fig. 9. Comparison between the new SA Vs the classic SA (τ = 0.8)

Conclusion

In the present paper a new cooling law has been proposed for an optimization procedure based on the use of the SA algorithms. Such a law has been tested on two different kind of problem taking into account two cases study already treated in the literature. The results show that its utilization allows at obtaining a very higher convergence quickness than that one arising from the use of the traditional cooling law of the SA algorithms.

Assuming as valid the hypothesis about the only parameter involved in the new formulation, the implementation of algorithms based on its use does not require any previous tuning phase so determining a further reduction of the global run-time.

Funding Information

This work was supported by the Università degli Studi di Palermo.

Author's Contributions

This study is the result of the full and equal collaboration of all the authors.

Ethics

The authors have no conflicts of interest in the development and publication of the this research.

References

Aarts, E. and J. Korst, 1989. Simulated Annealing And Boltzmann Machines: A Stochastic Approach to Combinatorial Optimization and Neural Computing.1st Edn., John Wiley and Sons, ISBN-10: 0471921467, pp: 272.

Aghaie, A. and H. Mokhtari, 2009. Ant colony optimization algorithm for stochastic project crashing problem in PERT networks using MC simulation. Int. J. Adv. Manufacturing Technol., 45: 1051-1067. DOI: 10.1007/s00170-009-2051-6

Arikan, F. and Z. Gungor, 2001. An application of fuzzy goal programming to a multiobjective project network problem. Fuzzy Sets Syst., 119: 49-58. DOI: 10.1016/S0165-0114(99)00119-0

Azizi, N. and S. Zolfaghari, 2004. Adaptive temperature control for simulated annealing: A comparative study. Comput. Operat. Res., 31: 2439-2451. DOI: 10.1016/S0305-0548(03)00197-7

Cheng, R., M. Gen and Y. Tsujimura, 1996. A tutorial survey of job-shop scheduling problems using genetic algorithms-I representation. Comput. Industrial Eng., 30: 983-997. DOI: 10.1016/0360-8352(96)00047-2

Cheung, W. and H. Zhou, 2002. Using genetic algorithms and heuristics for job shop scheduling with sequence-dependent setup times. Annals Operat. Res., 107: 65-81. DOI: 10.1023/A:1014990729837

Dowsland, K., 1993. Some experiments with simulated annealing techniques for packing problems. Eur. J. Operat. Res., 68: 389-399. DOI: 10.1016/0377-2217(93)90195-S

Feng, C.W., L.Y. Liu and S.A. Burns, 1997. Using genetic algorithms to solve construction time-cost trade-off problems. J. Comput. Civil Eng., 11: 184-189. DOI: 10.1061/(ASCE)0887-3801(1997)11:3(184)

Geman, S. and D. Geman, 1984. Stochastic relaxation, Gibbs distribution and the Bayesian restoration of images. IEEE Trans. Patt. Analysis Machine Intellig., 6: 721-741. DOI: 10.1109/TPAMI.1984.4767596

Hajek, B., 1988. Cooling schedules for optimal annealing. Math. Operat. Res., 13: 311-329. DOI: 10.1287/moor.13.2.311

Hertz, A. and M. Widmer, 1996. An improved tabu search approach for solving the job shop scheduling problem with tooling constraints. Discrete Applied Math., 65: 319-345. DOI: 10.1016/0166-218X(95)00040-X

Huang, M., F. Romeo and A. Sangiovanni-Vincetelli, 1986. An efficient general cooling schedule for simulated annealing. Proceedings of the IEEE International Conference on Computer Aided Design, (Nov. 11-13), Santa Clara, CA. pp: 381-384.

Jin, F., S. Song and C. Wu, 2009. A simulated annealing algorithm for single machine scheduling problems with family setups. Comput. Operat. Res., 36: 2133-2138. DOI: 10.1016/j.cor.2008.08.001

Kia, R., A. Baboli, N. Javadian, R. Tavakkoli-Moghaddam and M. Kazemi *et al*., 2012. Solving a group layout design model of a dynamic cellular manufacturing system with alternative process routings, lot splitting and flexible reconfiguration by simulated annealing. Comput. Operat. Res. DOI: 10.1016/j.cor.2012.01.012

Kim, S.C. and P.M. Bobrowski, 1997. Scheduling jobs with uncertain setup times and sequence dependency. Omega, 25: 437-447. DOI: 10.1016/S0305-0483(97)00013-3

Kirkpatrick, S., C.D. Gelatt and M.P. Vecchi, 1983. Optimization by simulated annealing. Science, 220: 671-680. DOI: 10.1126/science.220.4598.671

Leu, S.S. and C.H. Yang, 1999. GA-based multicriteria optimal model for construction scheduling. J. Construct. Eng. Manage., 125: 420-427. DOI: 10.1061/(ASCE)0733-9364(1999)125:6(420)

Leung, S.C.H., D. Zhang, C. Zhou and T. Wu, 2012. A hybrid simulated annealing metaheuristic algorithm for the two-dimensional knapsack packing problem. Comput. Operat. Res., 39: 64-73. DOI: 10.1016/j.cor.2010.10.022

Li, H. and P. Love, 1997. Using improved genetic algorithms to facilitate time-cost optimization. J. Construct. Eng. Manage., 123: 233-237. DOI: 10.1061/(ASCE)0733-9364(1997)123:3(233)

Li, H., J.N. Cao and P. Love, 1999. Using machine learning and GA to solve time-cost trade-off problems. J. Construct. Eng. Manage., 125: 347-353. DOI: 10.1061/(ASCE)0733-9364(1999)125:5(347)

Liberatore, M.J. and B. Pollack-Johnson, 2006. Extending project time-cost analysis by removing precedence relationships and task streaming. Int. J. Project Manage., 24: 529-535. DOI: 10.1016/j.ijproman.2006.04.004

Lin, S.W. and K.C. Ying, 2013. Minimizing makespan and total flowtime in permutation flowshops by a bi-objective multi-start simulated annealing algorithm. Comput. Operat. Res. DOI: 10.1016/j.cor.2011.08.009

Lin, S.W., J.N.D. Gupta, K.C. Ying and Z.J. Lee, 2009. Using simulated annealing to schedule a flowshop manufacturing cell with sequence-dependent family setup times. Int. J. Product. Res., 47: 3205-3217. DOI: 10.1080/00207540701813210

Low, C., 2005. Simulated annealing heuristic for flow shop scheduling problems with unrelated parallel machines. Comput. Operat. Res., 32: 2013-2025. DOI: 10.1016/j.cor.2004.01.003

Luh, P.B., L. Gou, Y. Zhang, T. Nagahora and M Tsuji *et al*., 1998. Job shop scheduling with group-dependent setups, finite buffers and long time horizon. Annals Operat. Res. Math. Industrial Syst., 76: 233-259. DOI: 10.1023/A:1018948621875

Lundy, M. and A. Mees, 1986. Convergence of an annealing algorithm. Math. Programm., 34: 111-124. DOI: 10.1007/BF01582166

Mason, S.J., J.W. Fowler, W.M. Carlyle and D.C. Montgomery, 2005. Heuristics for minimizing total weighted tardiness in complex job shops. Int. J. Product. Res., 43: 1943-1963. DOI: 10.1080/00207540412331331399

Metropolis, N., A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller and E. Teller, 1953. Equation of state calculations by fast computing machines. J. Chem. Phys., 21: 1087-1092. DOI: 10.1063/1.1699114

Michalewicz, Z., 1999. Genetic Algorithms + Data Structures = Evolution Programs. 1st Edn., Springer Science and Business Media, Berlin, ISBN-10: 3540606769, pp: 387.

Pinedo, M., 2002. In Englewood Cliffs. NJ: Prentice Hall. Scheduling: Theory, Algorithms and Systems: 2nd Edn.

PMI, 2013. A Guide to the Project Management Body of Knowledge (PMBOK® Guide). 5th Edn., Project Management Institute, Pennsylvania, ISBN-10: 1935589679, pp: 589.

Rong-Ceng, L., 2006. A new method for unit maintenance scheduling considering reliability and operation expense. Electrical Power Energy Syst., 28: 471-481. DOI: 10.1016/j.ijepes.2006.02.009

Sitarz, S., 2009. Ant algorithms and simulated annealing for multicriteria dynamic programming. Comput. Operat. Res., 36: 433-441. DOI: 10.1016/j.cor.2007.09.011

Storer, R.H., S.D. Wu and R. Vaccari, 1992. New search spaces for sequencing problems with application to job shop scheduling. Manage. Sci., 38: 1495-1509. DOI: 10.1287/mnsc.38.10.1495

Sun, J.U., S.R. Yee and H. Hwang, 2003. Job shop scheduling with sequence dependent setup times to minimize makespan. Int. J. Industrial Eng. Theory Applicat. Practice, 10: 455-461.

Timur, K., Y. Mehmet and B. Mehmet, 2012. An ant colony optimization algorithm for load balancing in parallel machines with sequence-dependent setup times. Comput. Operat. Res., 39: 1225-1235. DOI: 10.1016/j.cor.2010.12.003

Tung, Y.I., 2007. Performing complex project crashing analysis with aid of particle swarm optimization algorithm. Int. J. Project Manag., 25: 637- 646. DOI: 10.1016/j.ijproman.2006.11.001

Xinchao, Z., 2010. Simulated annealing algorithm with adaptive neighborhood. Applied Soft Comput. J., 11: 1827-1836. DOI: 10.1016/j.asoc.2010.05.029

Zahrani, M.S., M.J. Loomes, J.A. Malcolm, A.Z.M. Dayem Ullah and K. Steinhofel *et al*., 2008. Genetic local search for multicast routing with pre-processing by logarithmic simulated annealing. Comput. Operat. Res., 35: 2049-2070. DOI: 10.1016/j.cor.2006.10.001

Zhu, X. and W.E. Wilhelm, 2006. Scheduling and lot sizing with sequence-dependent setup: A literature review. IIE Transact., 38: 987-1007. DOI: 10.1080/07408170600559706