Simulated Annealing with Restart to Job Shop Scheduling Problem Using Upper Bounds

Marco Antonio Cruz-Chavez[1] and Juan Frausto-Solis[2]

[1] Faculty of Chemical Sciences and Engineering, Autonomous University of Morelos State Av. Universidad 1001, Col. Chamilpa, 62270, Cuernavaca, Morelos, MÉXICO
mcruz@buzon.uaem.mx

[2] Department of Computer Science, ITESM, Campus Cuernavaca Paseo de la Reforma 182-A, 62589, Temixco, Morelos, MÉXICO
juan.frausto@itesm.mx

Abstract. An algorithm of simulated annealing for the job shop scheduling problem is presented. The proposed algorithm restarts with a new value every time the previous algorithm finishes. To begin the process of annealing, the starting point is a randomly generated schedule with the condition that the initial value of the makespan of the schedule does not surpass a previously established upper bound. The experimental results show the importance of using upper bounds in simulated annealing in order to more quickly approach good solutions.

1 Introduccion

The job shop scheduling problem (JSSP) is considered to be one of the most difficult to solve in combinatorial optimization. It is also one of the most difficult problems in the NP-hard class [1].

The JSSP consists of a set of machines that each carry out the execution of a set of jobs. Each job consists of a certain number of operations, which must be carried out in a specific order. Each operation is carried out by a specific machine and has a specific time of execution. Each machine can execute a maximum of one operation at any given point in time. A single machine is unable to carry out more than one operation of the same job. The objective of the problem is to find the makespan. The makespan is defined as the time it takes to complete the last operation in the system. An immense number of models exist that represent the JSSP, but the two most important and influential models are those of disjunctive formulation [2] and disjunctive graph [2] and [3].

The simulated annealing algorithm (SA) [4] is an analogy between the annealing process of solids and the problem of solving combinatorial optimization problems. This algorithm has been used with high rates of success for JSSP by several researchers [5], [6], [7], [8], [9], and [10]. For the JSSP, S is a schedule obtained by using a randomly generated initial point. S' is in the neighborhood of S, which is obtained by a small perturbation of S. T_0 and T_f are the intial and final temperatures of the process. β is the coefficient of temperature that controls the cooling of the system. $f(S)$ is the energy of the configuration S, which is

generally the makespan. One of the ways of perturbing the neighborhood of S is proposed by Balas [3], and involves exchanging a pair of adjacent operations that are within critical blocks of operations. This form of altering the neighborhood is known as N_1. The critical blocks of operations are the operations that form the longest path of the schedule. Each critical block of operations that form this path are performed by a common machine. N_1, have been used previously in SA with good results by [5], [6], [7], [8], and [10]. N_1 is what is used in this work due to ease of implementation.

Other researchers have developed variations of N_1. The algorithm of Matsuo et al. [7] is a derivation of N_1, called N_{1a}. This type of change to the neighborhood involves changing the placement of three pairs of adjacent operations simultaneously, where each operation pair is performed by a different machine. The algorithm of Aart et al. [5], also a derivation of N_1, called N_{1b}, involves reversing three adjacent pairs of operations that are all performed by the same machine, and with the condition that one of the pairs does not form the longest path.

Another type of derivation of N_1 is the neighborhood of critical block (CB), which is called N_2. In this type of neighborhood, one operation in the block is changed for either the initial or final operation of the block. It is not required that the operations that change places be adjacent. The algorithms that use N_2 are the CBSA of Yamada et al. [10] and the CSBA+CB of Yamada and Nakano [9]. This last algorithm uses a deterministic local search called shifting bottleneck (SB) [11]. It uses SB when the schedule S', which is a perturbation of S, is rejected in the SA. When the rejection occurs, the shifting bottleneck is applied to S' in order to improve it. Once S' is improved, although $f(S')$ come to be greater than $f(S)$, the solution is accepted. In this type of neighborhood, the shifting bottleneck must be used whenever S' is rejected in the algorithm. The implementation of the SB is not easy because it requires that the release time and due dates are calculated. Here, an algorithm is proposed which is easy to implement and produces high quality solutions; it is a simulated annealing algorithm with restart [12].

2 Simulated Annealing with Restart Using Upper Bounds

The proposed algorithm of simulated annealing with restart (SAR) consists of executing a set of SA. The SAR algorithm using upper bound can be seen in Figure 1. Each simulated annealing that is executed involves the iteration of the SAR algorithm. Each repetition begins using a different schedule. The idea of beginning with different schedules for each repetition of SAR came about because of experimental tests that were carried out. In the experimental tests, it was detected that even though SA allows the search for a global optimum, at some point low temperatures eliminate this possibility and the search finds a local optimum. By beginning with different schedules, more variability of solutions is obtained than by beginning with only a single schedule. It is seen that the restart

1. Given initial iteration $k = 0$, initial values of S_f, T_f, β
2. Beginning of annealing $k = k + 1$:
  3. $S = S_0 \leq Upper\ \ Bound$, $T = T_0$, initial S_c
  4. While the final temperature Tf is not reached,
    5. While equilibrium is not reached:
      •Generate a state $S\prime$ by means of a perturbation in S
      •if $f(S\prime) - f(S) \leq 0$ then $S = S\prime$
      •if $f(S\prime) - f(S) > 0$ the state is accepted with
        the probability: $P_{accept} = e^{-\left(\frac{f(S\prime)-f(S)}{T}\right)}$
      •With a randomly generated number α
        evenly distributed between $(0, 1)$
      •if $\alpha < P_{accept}$ then $S = S\prime$
      •if $S < S_c$ then $S_c = S$
      •If the equilibrium does not exist, return to 5
   $T = T * \alpha$
  The best configuration is stored, if $S_c < S_f$ then $S_f = S_c$
  If $T \geq T_f$, return to 4
If $k < maxiter$, return to 2 to begin a new annealing
The solution is S_f

Fig. 1. The simulated annealing algorithm with restart to JSSP using an upper bound

of the simulated annealing algorithm leaves a different point in the solution space each time it is repeated.

The restarting of the algorithm allows for the search of global optimums by using a new schedule in each repetition of SAR. This allows for a different part of the solution space to be explored when SAR is at a local optimum. This not only increases the probability of finding a global optimum, but also increases the time of the search. In the SAR algorithm, at the beginning of each simulated annealing, an UB (Upper Bound) is established in order to randomly obtain the schedule with which to start the process. The value of the makespan of this schedule may not be greater than the UB, or the schedule is not accepted as the initial configuration of the annealing. The UB was established by trial and error, so that the algorithm took no longer than 15 seconds to find an I-SCHED (initial schedule) that could be used to begin the repetitions of the SAR algorithm. In order to obtain an I-SCHED that did not surpass the UB, a random procedure was used. First, a random sequence of operations was formed for each machine. Next, a scheduling algorithm [13] was used to eliminate global cycles. Finally, this schedule is improved by using the Giffler and Thompson algorithm [14] that obtains active schedules [6]. With the proposed procedure, obtaining good initial solutions that they don't surpass an UB in a short period of time could be assured.

In each iteration of the SAR algorithm, SA is fully completed. The best configuration after all the repetitions are completed is the final solution. The number

of annealings carried out, that is, the number of repetitions of the algorithm, is a function of time of execution and depends on the problem.

Other form of restart with SA is proposed by Yamada and Nakano [9], Yamada et al. [10] and Aydin and Fogarty [15].

3 Computational Results

The proposed algorithm was proven with two benchmark registered in the OR library [16], FT10 of 10x10 and LA40 of 15x15. For the FT10, ten trials were done with an UB = 1570, generating the initial schedules with the procedure I-SCHED, and ten trials were done without an UB, but also using the procedure of I-SCHED. The parameters of $T_0 = 32*$(Makespan of I-SCHED), $T_f = 1.0$, $\beta = 0.98$, and N_1 as the type of neighborhood were equal in each test performed, whether there was an UB or not. In both cases, a maximum time of four hours was used to obtain the results. When the UB is used, the optimum (930) is obtained in 80% of trials, the quickest being obtained in 44 minutes, 55 seconds. The standard deviation is 2.95 with an average makespan of 931.4. If an UB is not used, the obtained results are of poor quality (the optimum could not be obtained), with a standard deviation of 4.97 and an average makespan of 941.9. We also compared the results obtained with the algorithms ASSA using N_1 and CBSA N_2, of Yamada et al. [10] because they have obtained some of the best results for the problem FT10. They also carried out ten trials. It can be seen that ASSA presents a greater standard deviation of 5.10 and a higher average makespan of 939.5. The worst result obtained by the SAR algorithm for the makespan was 937 and by ASSA 951. It is obvious that the SAR algorithm is more effective than ASSA in obtaining the optimum for the problem FT10. For the CBSA algorithm, in ten trials, and average makespan of 930.8 was obtained and a standard deviation of 2.4. Only in one trial was CBSA unable to obtain the optimum result, this trial gave a makespan of 938. These results indicate that CBSA, by a small margin, is better able to find accurate solutions than SAR. It is believed that the CBSA obtains better results due to the neighborhood it uses, because the only difference between ASSA and CBSA is the type of neighborhood used.

It is important to emphasize that a great number of tests were generated, with sequences of slower cooling in both cases, with and without UB. The results of all the performed tests slower cooling were farther from the global optimum. In addition, there was a considerable increase in the time of execution of the algorithm in order to find these solutions.

Table 1 shows the results of several algorithms of SA for the problem FT10. In the table, the type of neighborhood each author used is specified. None of these algorithms involve restarting the annealing so their time of execution is small. It can also be observed that most of the results are poorer than those obtained by the SAR when an upper bound is not applied. Table 2 shows the best performance of several algorithms of SA, including the SAR algorithm, for the benchmark LA40 of JSSP.

Table 1. Results of several simulated annealing algorithms for the problem FT10

FT10 , 10 x 10, optimum = 930		
Authors	$t = seg$	Makespan
Aart et al. (N_1)	99	969
Aart et al. (N_{1b})	99	977
Van Laarhoven (N_1)	3895	951
Matsuo et al. (N_{1a})	987	946

Table 2. Results of several simulated annealing algorithm for the problem LA40

LA40, 15 X 15, optimum = 1222		
Algorithm	Makespa	%ER
CBSA+SB(N_2)	1228	0.49
SAR(N_1)	1233	0.90
VanLaarhoven(N_1)	1234	0.98
Matsuo et al.(N_{1a})	1235	1.06
CBSA(N_2)	1235	1.06
Aart et al.(N_{1b})	1254	2.62
Aart et al.(N_1)	1256	2.78

The parameters in SAR for LA40 were fixed to: UB = 2300, $T_0 = 25$, $T_f = 5.0$ and $\beta = 0.99$. In the same table 2 it can be observed that the result obtained by SAR with a 0.90% relative error is better than all of the other algorithms that use N_1 and derivatives of N_1. SAR also surpasses the CBSA N_2. SAR is surpassed only by CBSA+SB. The CBSA+SB algorithm use the neighborhood N_2 and implements the procedure of deterministic local search, called shifting bottleneck, for the re-optimization of schedules obtained in each repetition of the algorithm.

4 Conclusion

The use of upper bounds in the algorithm allows the solution of the problem FT10 to be found in almost all trials. This indicates that for this problem, in the SAR algorithm, starting with the good schedules that do not surpass an UB, improves the solution considerably. Also, it is recommended that the simulated annealing be restarted in several points with good schedules. By doing this, better solutions are obtained which are nearer to the global optimum.

The quality of the solution of SAR (N_1) for the problem LA40 is comparable to the CBSA+SB (N_2). One advantage that SAR has over the CBSA+SB is that for the CBSA+SB, it is necessary to find the machines with the shifting bottleneck in each repetition. To find all of this information, it is necessary to calculate the release times and due dates of each operation that is involved in

the problem. Thus, because of the similar quality and simpler implementation, SAR appears to be of more interest from a scientific point of view. It is thought that SAR would have better performance with large problems than CBSA+SB due to the fact that SAR does not use deterministic local search procedures. Better performance would be possible in SAR using N_2.

References

1. Garey, M.R., Johnson, D.S. and Sethi, R., The complexity of Flow shop and Job shop Scheduling. Mathematics of Operations Research, Vol. I, No 2, USA, 117-129, May, 1976.
2. Conway, R. W., Maxwell, W.L. and Miller, L. W., Theory of Scheduling, Addison-Wesley, Reading, Massachusetts, 1967.
3. Balas, E., Machine sequencing via disjunctive graphs: an implicit enumeration algorithm, Oper. Res., 17:941-957, 1969.
4. Kirkpatrick, S., Gelatt Jr., S. D. and Vecchi, M. P., Optimization by simulated anneal-ing. Science, 220(4598), 13 May, 671-680, 1983.
5. Aarts, E.H.L., Van Laarhoven, P.J.M., Lenstra, J.K. and Ulder, N.L.J., A computational study of local search algorithms for job shop scheduling, ORSA Journal on Computing 6, 118-125, 1994.
6. Pinedo, M., Scheduling Theory, Algorithms, and Systems, Prentice Hall, U.S.A., 1995.
7. Matsuo, H., Suii, C.J. and Sullivan, R.S., A controlled search simulated annealing method for the general job shop scheduling problem, Working paper 03-04-88, Graduate School of Business, University of Texas, Austin, 1988.
8. Van Laarhoven, P.J.M. , Aarts, E.H.L. and Lenstra, J.K., Job shop scheduling by simulated annealing. Oper. Res., 40(1):113-125, 1992.
9. Yamada, T., and Nakano, R., Job-shop scheduling by simulated annealing combined with deterministic local search, Meta-heuristics: theory and applications, Kluwer academic publishers MA, USA, pp. 237-248, 1996.
10. Yamada, T., Rosen, B. E. and Nakano, R., A simulated annealing approach to job shop scheduling using critical block transition operators, IEEE, 0-7803-1901-X/94, 1994.
11. Adams, J., Balas, E. and Zawack, D., The shifting bottleneck procedure for job shop scheduling, Mgmt. Sci., 34, 1988.
12. Ingber, L., Simulated annealing: Practice versus theory, Mathematical Computer Modelling, 18(11), 29-57, 1993.
13. Nakano, R. and Yamada, T., Conventional Genetic Algorithm for Job-Shop. Problems, in Kenneth, M. K. and Booker, L. B. (eds) Proceedings of the 4th International Conference on Genetic Algorithms and their Applications, San Diego, USA, pp. 474-479, 1991.
14. Zalzala, P. J. and Flemming, Zalsala, A.M.S. (Ali M.S.), ed., Genetic algorithms in engineering systems /Edited by A.M.S. Institution of Electrical Engineers, London, 1997.
15. Aydin, M. E. and Fogarty, T. C., Modular Simulated annealing algorithm for job shop scheduling running on distributed resource machin (DRM), South Bank University, SCISM, 103 Borough Road, London, SE1 0AA, UK, 2002.
16. Beasley, J.E., OR Library, Imperial College, Management School, http://mscmga.ms.ic.ac.uk/info.html, 1990.