Integrationsaspekte der Simulation:
Technik, Organisation und Personal
Gert Zülch & Patricia Stock (Hrsg.)
Karlsruhe, KIT Scientific Publishing 2010

Using simulation to Provide Alternative Solutions to the flowshop sequencing problem

Einsatz von Simulation zur Generierung alternativer Lösungen für das Flowshop Sequencing Problem

Angel Juan, Antoni Guix, Ferran Adelantado
Open University of Catalonia, Barcelona (Spain)
ajuanp@uoc.edu

Pau Fonseca i Casas
UniversitatPolitècnica de Catalunya, Barcelona (Spain)
pau@fib.upc.edu

Ruben Ruiz
ITI-Universidad Politécnica de Valencia (Spain)

Abstract: In this paper we present SS-GNEH, a simulation-based algorithm for the Permutation Flowshop Sequencing Problem (PFSP). Given a PFSP instance, the SS-GNEH algorithm incorporates a randomness criterion to the classical NEH heuristic and starts an iterative process in order to obtain a set of alternative solutions, each of which outperforms the NEH algorithm. Thus, a random but oriented local search of the space of solutions is performed, and a list of "good alternative solutions" is obtained. We can then consider several desired properties per solution other than maximum time employed, such as balanced idle times among machines, number of completed jobs at a given target time, etc. This allows the decision-maker to consider multiple solution characteristics other than just those defined by the aprioristic objective function. Therefore, our methodology provides flexibility during the sequence selection process, which may help to improve the scheduling process. Several tests have been performed to discuss the effectiveness of this approach. The results obtained so far are promising enough to encourage further developments on the algorithm and its applications in real-life scenarios.

1 Introduction

The Permutation Flowshop Sequencing Problem (PFSP) is a well-known scheduling problem that can be formulated as follows: a set J of n independent jobs has to be processed on a set M of m independent machines. Every job $j \in J$ requires a given fixed processing time on every machine $i \in M$. This processing time can be denoted

as $p_{ij} \geq 0$. Each machine can execute at most one job at a time, and it is assumed that the jobs are processed by all machines in the same order they have been processed by the first machine. The classical goal here is to find a sequence for processing the jobs in the shop so that a given criterion is optimized. The criterion that is most commonly used is the minimization of the maximum completion time (makespan) of the production sequence. The described problem is usually denoted as Fm|prmu|Cmax, using the notation proposed by GRAHAM et al. (1979). It is a combinatorial problem with *n!* possible sequences which, in general, is NP-complete. Similar to what has happened with other combinatorial problems, a large number of different approaches have been developed to deal with the PFSP. These approaches range from the use of pure optimization methods, such as mixed integer programming, for solving small-sized problems, see REDDI and RAMAMOORTHY (1972), to the use of heuristics and meta-heuristics that provide near-optimal solutions for medium and large-sized problems, NAWAZ et al. (1983) or CHANDRASEKHARAN and ZIEGLER (2004).Most of these methods focus on minimizing makespan. However, utility functions of decision-makers are difficult to model and, frequently, criteria other than total time employed to process the list of jobs should be also considered in real-life scenarios. For that reason, there is a need for more flexible methods able to provide a large set of alternative near-optimal solutions with different properties, so that decision-makers can choose among different alternative solutions according to their specific necessities and preferences.

2 Literature Review

A large number of heuristics and metaheuristics have been proposed to solve the PFSP, mainly because of the difficulty encountered by exact methods to solve medium or large instances. As explained before, most existing approaches focus on the objective of minimizing makespan. NAWAZ et al. (1983) introduced the NEH heuristic (Nawaz, Enscore and Ham), which is commonly considered as the best performing heuristic for the PFSP. Basically, what the NEH heuristic proposes is to calculate the total processing time required for each job (i.e., the total time each jobs requires to be processed by the set of machines), and then to create an "efficiency list" of jobs sorted in a descending order according to this total processing time. At each step, the job at the top of the efficiency list is selected and used to construct the solution, that is: the "common sense" rule is to select first those jobs with the highest total processing time. Once selected, a job is inserted in the sorted set of jobs that are configuring the on-going solution. The exact position that the selected job will occupy in that ongoing solution is given by the minimizing makespan criterion. TAILLARD (1990) introduced a data structure that reduces the NEH complexity. Other interesting heuristics are those from SULIMAN (2000) or FRAMINAN and LEISTEN (2003), which also consider several extensions of NEH when facing other (single) objectives than makespan. Different metaheuristic approaches have been also proposed for the PFSP. Using Simulated Annealing an interesting work is OSMAN and POTTS (1989). A Tabu Search algorithm known as SPIRIT was proposed by WIDMER and HERTZ (1989). Other authors propose the use of Genetic Algorithms for solving the PFSP which were also based on the NEH heuristic (CHEN et al. 1995; REEVES 1995; ALDOWAISAN, ALLAHVEDI 2003). As examples of other works which also rely on the use of the NEH heuristic an interesting approach is the presented in the Ant Colony Optimization algorithm of

(CHANDRASEKHARAN, ZIEGLER 2004). All the aforementioned work has in common that the algorithms proposed are easy to code and therefore the results can be reproduced without too much difficulty. In addition, many of the above algorithms can be adapted to other more realistic flowshop environments. Additionally, there are other highly elaborated hybrid techniques for solving the PFSP. However, as RUIZ and STÜTZLE (2007) point out, "they are very sophisticated and an arduous coding task is necessary for their implementation". In other words, it is unlikely that they can be used for solving realistic scenarios without direct support from the researchers that developed them.

3 Our Approach

The approach presented in this paper aims to provide a simple probabilistic algorithm which will be able to improve the results provided by the NEH heuristic in just a few iterations. Also, this algorithm should avoid complex or time-consuming fine-tuning processes. As explained before, a second goal of our approach is to provide not only a good solution, but a set of alternative solutions. In order to meet those goals, we decided to combine Monte Carlo Simulation (MCS) techniques with the NEH heuristic. In particular, our approach introduces a special random behaviour within the NEH heuristic and then starts an iterative process. This random behaviour helps us to start a search process inside the space of feasible solutions. Each of these feasible solutions will consist of a list of sorted jobs. As explained before, the NEH heuristic is an iterative algorithm which uses a sorted list of jobs to construct a solution for the PFSP. At each step of this iterative process, the NEH chooses the job which is at the top of that list – jobs are sorted in the list according to the total time they require to be processed by all the machines. As a result, the NEH provides a "common sense" deterministic solution. Our approach, instead, assigns a probability of selecting each job in the jobs list. According to our design, this probability should be coherent with the total time that each job needs to be processed by all machines, i.e., jobs with higher total times will be more likely to be selected from the list than those with lower total times. Finally, this selection process should be done without introducing too many parameters in the methodology – otherwise, it would be necessary to perform fine-tuning processes, which tend to be non-trivial and time-consuming. To reach all those goals, we employ the geometric statistical distribution with parameter α $(0 < \alpha < 1)$ during the solution-construction process (JUAN et al. 2010): each time a new job has to be selected from the list, a geometric distribution is randomly selected. This distribution is then used to assign exponentially diminishing probabilities to each eligible job according to its position inside the list, which has been previously sorted by its corresponding total-processing-time value. That way, jobs with higher processing times are always more likely to be selected from the list, but the probabilities assigned are variable and they depend upon the concrete distribution selected at each step. By iterating this procedure, an oriented random search process is started. Notice that this general approach has many similarities with the Greedy Randomized Adaptive Search Procedure (GRASP; FEO, RESENDE 1995). To some extent, our approach could be considered as part of the large family of Hybrid GRASP algorithms. Figure 1 illustrates the basics of our approach, which are detailed next. (i) Given a PFSP instance, construct the corresponding data model and use the classical NEH algorithm to solve it. (ii) Choose a probability distribution

(e.g. a geometric or a zipf) for adding random behaviour to the algorithm. (iii) Start an iterative process to generate solutions using the Generalized NEH for the Scheduling flowshop Sequence problem (SS-GNEH) algorithm. (iv) For each iteration, save the resulting solution in a database only if it outperforms the one provided by the NEH algorithm, i.e., we will consider that a solution is a good one only if it outperforms the NEH solution from a makespan perspective.

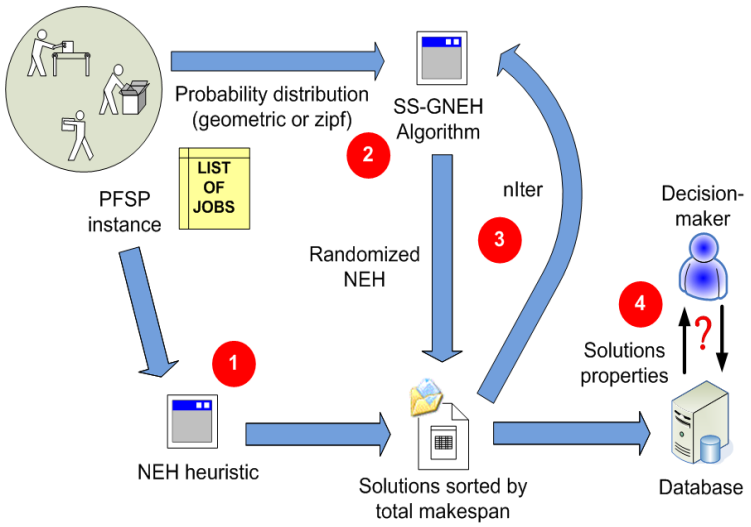

***Figure 1**: Basic schema of our approach*

Figure 2 shows the main procedure of the SS-GNEH algorithm, which drives the solving methodology. Roughly speaking, if the size of the jobs list in the current step is large enough, the parameter α can be interpreted as the probability of selecting the job with the highest total-processing-time value at the current step of the solution-construction process.

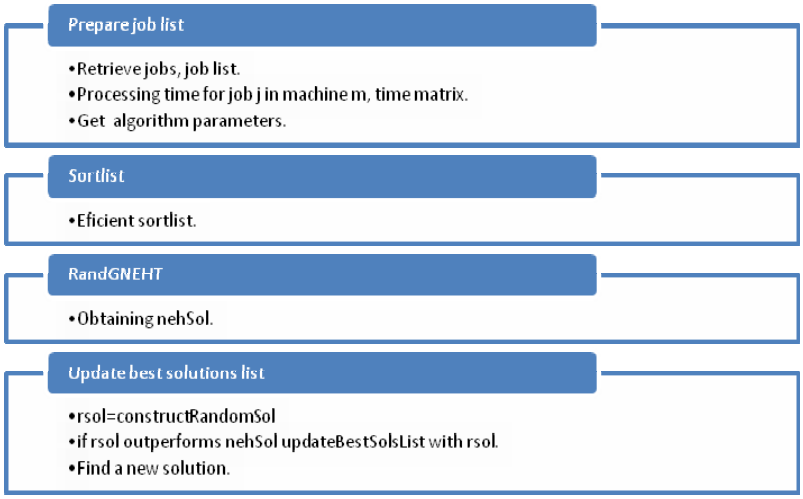

***Figure 2:** SS-GNEH main procedure*

Choosing a relatively low α-value (e.g. $\alpha = 0.05$) implies considering a large number of jobs from the list as potentially eligible. On the contrary, choosing a relatively high α-value (e.g. $\alpha = 0.25$) implies reducing the list of potential eligible jobs to just a few of them. Once a value for α is chosen, the first job must be selected. This same value of α can be used for all future steps. In that case, the selection of the α-value might require some minor fine-tuning process. However, based on the tests we have performed, we prefer to consider this α-value as a random variable whose behaviour is determined by a well-known continuous distribution, e.g.: a uniform distribution in the interval *(0.05, 0.25)*. This way, we do not only avoid a fine-tuning process, but we also have the possibility to combine different values of this parameter at different job-selection steps of the same solution-construction process.

4 Experimental Results

The methodology described has been implemented as a Java application that runs directly on the Eclipse IDE for Java. A standard personal computer, Intel® Core™2 Duo CPU at 2.4 GHz and 2 GB RAM, was used to perform all tests. We tested the first 60 benchmark instances from TAILLARD (1993). We performed 15 executions for each instance. These instances can be found at http://mistic.heig-vd.ch/taillard/default.htm. For each tested instance, we also considered its associated best-known solution (BKS) as reported in ZOBOLAS et al. (2009). Table 1 shows the results for the first 30 Taillard's instances, which consider 20 jobs cases. Notice the significant reduction in the average gap with respect to the best-know solution (BKS), from *3.86 %* (NEH) to *0.01 %* (SS-GNEH). Table 1 shows that for the first 30 Taillard's instances our methodology is able to provide either the optimal solution or an alternative one (gap < *0.01 %*). Regarding the reported CPU times, the average time for solving these instances is about 1 second. These times can be easily improved by just coding the algorithm in C/C++ instead of Java. Table 2 shows results for the Taillard's instances with 50 jobs. These are larger instances but again our methodology provides a significant reduction in the average gap (from *4.15 %* to *1.44 %*) in a reasonable amount of computing time. According to the experimental tests carried out, each iteration is completed in just a few milliseconds. By construction, odds are than the generated solution outperforms the one given by the NEH heuristic. This means that our approach provides, after a few iterations, a feasible solution which outperforms the NEH heuristic as regards makespan. Moreover, as verified by testing, several alternative solutions improving the NEH can be obtained after some more iterations (which take more CPU time), each of them having different attributes regarding criteria such as balanced idle times among machines, number of completed jobs at a given target time, etc.

5 Conclusions

In this paper a simple probabilistic methodology for solving the Permutation Flowshop Sequencing Problem (PFSP) has been presented. This methodology, which does not require any particular fine-tuning or configuration process, combines the classical NEH heuristic with simulation using a geometric distribution. An important point to consider is the simplicity of the presented methodology. In effect, our algorithm needs little instantiation and does not require any fine-tuning or set-up

processes. This is quite interesting, since as it was noticed before, some of the most efficient meta-heuristics are not used in practice because of the difficulties they present when trying to implementing them. On the contrary, simple hybrid approaches like the one introduced here tend to be more flexible and, therefore, they seem more appropriate to deal with real restrictions and dynamic work conditions. Results show that our methodology is able to improve the NEH heuristic in just a few iterations. Moreover, being a constructive approach, it can generate several alternative good solutions in a reasonable time-period.

Taillard's Instance	# Jobs	# Machines	NEH Solution	Best-Known Solution (BKS)	Gap BKS-NEH	SS-GNEH Solution	Gap BKS-SS-GNEH	Running Time (mm:ss)
1	20	5	1286	1278	0.63%	1278	0.00%	00:00
2	20	5	1365	1359	0.44%	1359	0.00%	00:00
3	20	5	1159	1081	7.22%	1081	0.00%	00:00
4	20	5	1325	1293	2.47%	1293	0.00%	00:00
5	20	5	1305	1235	5.67%	1235	0.00%	00:00
6	20	5	1228	1195	2.76%	1195	0.00%	00:00
7	20	5	1278	1239	3.15%	1239	0.00%	00:00
8	20	5	1223	1206	1.41%	1206	0.00%	00:00
9	20	5	1291	1230	4.96%	1230	0.00%	00:00
10	20	5	1151	1108	3.88%	1108	0.00%	00:00
11	20	10	1680	1582	6.19%	1582	0.00%	00:00
12	20	10	1729	1659	4.22%	1659	0.00%	00:06
13	20	10	1557	1496	4.08%	1496	0.00%	00:02
14	20	10	1439	1377	4.50%	1377	0.00%	00:05
15	20	10	1502	1419	5.85%	1419	0.00%	00:00
16	20	10	1453	1397	4.01%	1397	0.00%	00:03
17	20	10	1562	1484	5.26%	1484	0.00%	00:00
18	20	10	1609	1538	4.62%	1538	0.00%	00:09
19	20	10	1647	1593	3.39%	1593	0.00%	00:02
20	20	10	1653	1591	3.90%	1591	0.00%	00:01
21	20	20	2410	2297	4.92%	2297	0.00%	00:08
22	20	20	2150	2099	2.43%	2099	0.00%	00:02
23	20	20	2411	2326	3.65%	2328	0.09%	00:05
24	20	20	2262	2223	1.75%	2223	0.00%	00:00
25	20	20	2397	2291	4.63%	2296	0.22%	00:04
26	20	20	2349	2226	5.53%	2226	0.00%	00:01
27	20	20	2362	2273	3.92%	2273	0.00%	00:05
28	20	20	2249	2200	2.23%	2202	0.09%	00:04
29	20	20	2320	2237	3.71%	2237	0.00%	00:00
30	20	20	2277	2178	4.55%	2178	0.00%	00:05
Averages					*3.86%*		*0.01%*	*00:01*

***Table 1**: Results for Taillard's instances 1 to 30 (20 jobs)*

Taillard's Instance	# Jobs	# Machines	NEH Solution	Best-Known Solution (BKS)	Gap BKS-NEH	SS-GNEH Solution	Gap BKS-SS-GNEH	Running Time (mm:ss)
31	50	5	2733	2724	0.33%	2724	0.00%	00:00
32	50	5	2843	2834	0.32%	2838	0.14%	00:00
33	50	5	2640	2621	0.72%	2621	0.00%	00:00
34	50	5	2782	2751	1.13%	2751	0.00%	00:00
35	50	5	2868	2863	0.17%	2863	0.00%	00:00
36	50	5	2850	2829	0.74%	2829	0.00%	00:01
37	50	5	2758	2725	1.21%	2725	0.00%	00:00
38	50	5	2721	2683	1.42%	2683	0.00%	00:00
39	50	5	2576	2552	0.94%	2552	0.00%	00:00
40	50	5	2790	2782	0.29%	2782	0.00%	00:00
41	50	10	3135	2991	4.81%	3055	2.14%	00:07
42	50	10	3032	2867	5.76%	2931	2.23%	00:02
43	50	10	2986	2839	5.18%	2905	2.32%	00:09
44	50	10	3198	3063	4.41%	3071	0.26%	00:01
45	50	10	3160	2976	6.18%	3034	1.95%	00:00
46	50	10	3178	3006	5.72%	3053	1.56%	00:04
47	50	10	3277	3093	5.95%	3142	1.58%	00:02
48	50	10	3123	3037	2.83%	3061	0.79%	00:09
49	50	10	3002	2897	3.62%	2934	1.28%	00:05
50	50	10	3257	3065	6.26%	3128	2.06%	00:06
51	50	20	4082	3850	6.03%	3958	2.81%	00:05
52	50	20	3921	3704	5.86%	3799	2.56%	00:06
53	50	20	3927	3640	7.88%	3755	3.16%	00:00
54	50	20	3969	3720	6.69%	3814	2.53%	00:03
55	50	20	3835	3610	6.23%	3711	2.80%	00:10
56	50	20	3914	3681	6.23%	3771	2.44%	00:03
57	50	20	3952	3704	6.70%	3800	2.59%	00:05
58	50	20	3938	3691	6.69%	3791	2.71%	00:08
59	50	20	3952	3743	5.58%	3838	2.54%	00:05
60	50	20	4079	3756	8.60%	3861	2.80%	00:06
Averages					4.15%		1.44%	00:03

***Table 2**: Results for Taillard's instances 31 to 60 (50 jobs)*

References

ALDOWAISAN, T.; ALLAHVEDI, A.: New heuristics for no-wait flowshops to minimize makespan. In: Computers and Operations Research, Oxford et al., 30(2003)8, pp. 1219-1231.

CHANDRASEKHARAN, R.; ZIEGLER, H.: Ant-colony algorithms for permutation flowshop scheduling to minimize makespan/total flowtime of jobs. In: European Journal of Operational Research, Amsterdam et al., 155(2004)2, pp. 426-438.

CHEN, C. L.; VEMPATI, V. S.; ALJABER, N.: An application of genetic algorithms for flow shop problems. In: European Journal of Operational Research, Amsterdam et al., 80(1995)2, pp. 389-396.

FEO, T. A.; RESENDE, M.: Greedy randomized adaptive search procedures. Journal of Global Optimization, Dordrecht et al., 6(1995)2, pp. 109-133.

FRAMINAN, J. M.; LEISTEN, R.: An efficient constructive heuristic for flowtime minimisation in permutation flow shops. In: OMEGA, Amsterdam et al., 31(2003)4, pp. 311-317.

GRAHAM, R. L.; LAWLER, E. L.; LENSTRA, J. K.; RINNOOY KAN, A. G. H.: Optimization and approximation in deterministic sequencing and scheduling: A survey. In: Annals of Discrete Mathematics, (1979), pp. 287-326

JUAN, A.; FAULIN, J.; RUIZ, R.; BARRIOS, B.; CABALLE, S.: The SR-GCWS hybrid algorithm for solving the capacitated vehicle routing problem. In: Applied Soft Computing, Amsterdam et al., 10(2010)1, pp. 215-224.

NAWAZ, M.; ENSCORE, E. E.; HAM, I.: A heuristic algorithm for the m-machine, n-job flowshop sequencing problem. In: OMEGA, Amsterdam et al., 11(1983)1, pp. 91-95.

OSMAN, L.; POTTS, C.: Simulated annealing for permutation flow-shop scheduling. In: OMEGA, Amsterdam et al., 17(1989)6, pp. 551-557.

REEVES, C. R.: A genetic algorithm for flowshop sequencing. In: Computers and Operations Research, Oxford et al., 22(1995)1, pp. 5-13.

REDDI, S. S.; RAMAMOORTHY, C. V.: On flowshop sequencing problem with nowaitin process. In: Operational Research Quarterly, (1972)23, pp. 323-331

RUIZ, R.; STÜTZLE, T.: A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. In: European Journal of Operational Research, Amsterdam et al., 177(2007)3, pp. 2033-2049.

SULIMAN, S.: A two-phase heuristic approach to the permutation flow-shop scheduling problem. In: International Journal of Production Economics, Amsterdam et al., 65(2000)1-3, pp. 143-152.

TAILLARD, E.: Some efficient heuristic methods for the flow shop sequencing problem. In: European Journal of Operational Research, Amsterdam et al., 47(1990)1, 65-74.

TAILLARD, E.: Benchmarks for basic scheduling problems. In: European Journal of Operations Research, Amsterdam et al., 64(1993)2, pp. 278-285.

WIDMER, M.; HERTZ, A.: A new heuristic method for the flow shop sequencing problem. In: European Journal of Operational Research, Amsterdam et al., 41(1989)2, pp. 186-193.

ZOBOLAS, C.; TARANTILIS, C.; IOANNOU, G.: Minimizing makespan in permutation flow shop scheduling problems using a hybrid metaheuristic algorithm. In: Computers and Operations Research, Oxford et al., 36(2009)4, 1249-1267.