A FILTER AND FAN APPROACH TO THE JOB SHOP SCHEDULING PROBLEM: A PRELIMINARY STUDY

Renato Duarte
Escola Superior de Ciências Empresariais - Instituto Politécnico de Setúbal
Campus do IPS, Estefanilha, 2910-503 Setúbal, Portugal
rduarte@esce.ips.pt

Cesar Rego
School of Business Administration and Hearin Center for Enterprise Science - University of Mississippi
University, MS 38677, USA
crego@bus.olemiss.edu

Dorabela Gamboa
Escola Superior de Tecnologia e Gestão de Felgueiras - Instituto Politécnico do Porto
Rua do Curral, Casa do Curral, Apt. 205, 4610-156 Felgueiras, Portugal
dgamboa@estgf.ipp.pt

Abstract – The Job Shop Scheduling Problem (JSSP) is a recognized difficult problem in combinatorial optimization for which extensive investigation has been devoted to the development of effective algorithms to find optimal or near-optimal solutions. In this paper we propose a new heuristic algorithm for the JSSP that effectively combines the well-known "shifting bottleneck" procedure (SBP) with a dynamic and adaptive neighborhood search procedure based on a "filter and fan" method (F&F). In the algorithm, SBP is used to generate a starting solution as well as to re-optimize the best schedules produced by the F&F method. The F&F is a local search procedure that generates compound moves in a tree search fashion. Preliminary results carried out on a set of 58 benchmark problems suggest that the marriage of the SBP with the F&F is a potentially good approach for the JSSP.

Keywords: heuristic methods, job shop scheduling, tree search, filter and fan.

I. Introduction

The Job Shop Scheduling Problem (JSSP) is a recognized difficult problem in combinatorial optimization. The problem is central in many supply chains that integrate logistics and production management. We consider the JSSP defined by a set of machines specialized to perform ordered operations unique for every job. No machine can perform more than one operation at a time, each operation has fixed time duration, and preemption is not allowed. The goal is to minimize the duration of the longest job in the schedule (i.e. the makespan). (For recent developments, see Nowicki and Smutnicki [16], and Grabowski and Wodecki [10].)

In this paper we propose a new heuristic algorithm for the JSSP that effectively combines the well-known Shifting Bottleneck Procedure (SBP) with a dynamic and adaptive neighborhood search procedure based on the Filter and Fan method (F&F). In the algorithm, SBP is used as a constructive method (that generates a starting solution) as well as a post-optimization procedure (that re-optimizes the best schedules produced by the F&F method). The F&F is a local search procedure that generates compound moves in a tree search fashion.

We provide a simple but fairly effective algorithm design and implementation to conduct a preliminary study on the potential of the F&F approach to the JSSP. The motivation for this study derives from the successful application of different types of compound neighborhoods to a wide range of difficult combinatorial optimization problems.

The most successful methods for generating compound neighborhoods are due to some special types of tree search neighborhoods and a variety of ejection chain methods. Perhaps, Beam Search [14] and the Lin-Kernighan [13] procedures are the most prominent examples of a tree search neighborhood and an ejection chain method, respectively. Although the application of the Lin-Kernighan procedure has been restricted to the traveling salesman setting, more general and advanced ejection chain methods have been the core of the most effective algorithms for a variety of major problems in combinatorial optimization, such as the traveling salesman [7, 8, 17], vehicle routing [18, 19], generalized assignment [23], and crew scheduling [5], just to cite a few.

Likewise, Beam Search has been typically used in the optimization of complex scheduling systems, including the JSSP [21]; however more advanced forms of tree search neighborhood approaches have been recently proposed and successfully applied to scheduling as well as to several other optimization problems. In particular, Balas and Vazacopoulus [4] consider a specialized neighborhood tree for the JSSP that led to one of the most effective algorithms for this problem. In fact, the success of this latter algorithm is the primary motivation for the study presented in this paper. Since the tree search neighborhood is the most fundamental difference between the Balas and Vazacopoulus's algorithm (B&V) and others not as much effective algorithms, it seems reasonable to assume that using more flexible and generalized forms of tree search would result in even more effective algorithms for the JSSP. Therefore, we consider the F&F method [9], which can be viewed as a natural generalization of Beam Search and which includes the B&V neighborhood tree as a special case. (For extensions of the method and recent applications, see Rego and Glover [20], and Greistorfer, Rego and Alidaee [11].)

The remainder of this paper is organized as follows. Section II describes the basic F&F model and Section III presents the proposed algorithm. Section IV discusses the experimental results obtained on a standard set of benchmark problems. Section V presents relevant concluding remarks and provides directions for further developments.

II. The Filter and Fan method

Filter and Fan (F&F) is a method to create *dynamic* and *adaptive* neighborhoods for local search and metaheuristic algorithms. The method is meant to generate compound moves made up of appropriate sequences of *simple* (or elementary) moves. The method proceeds as a tree search where branches represent elementary moves added at each level of the tree and nodes correspond to the *trial solutions* that result from the application of these moves. The *root* node represents the starting solution for the tree search. F&F is a breadth first search, since all the nodes are evaluated at each stage before going any deeper in the search tree.

Although the tree search procedure is the central part of the F&F search, the method that creates the root node can play an important role in the algorithm as well. The reason is that the F&F approach is conceived upon the creation of two candidate lists: a *filter candidate list* responsible for selecting candidate **solutions** at each level of the tree, and a *fan candidate list* used to determine candidate **moves** to operate on those solutions.

Each candidate list underlies the definition of appropriate criteria for the selection of their candidates as well as the legitimacy of their application to the solutions under consideration. In its simplest form the root node corresponds to a local optimum provided by any iterative local search procedure and the initial fan candidate list is typically made up of the highest evaluated moves in the neighborhood of the root node, and therefore created by the local search method that determined the local optimality. Again under this simplest setting, the filter candidate list may be limited to the selection of a subset of the solutions generated at each level that are the best in terms of the objective function value. However, more advanced strategies are meant to take advantage of adaptive memory as proposed in tabu search.

As the filter and fan candidate lists are both conceived to be dynamically updated the substitution of the local search method (that stops at first local optimum) by a tabu search procedure (that guides the search out of local optima) seems a natural extension to make effective use of adaptive memory programming.

In fact, note that although the F&F method provides a new strategy to effectively explore the neighborhood space by generating compound moves, the result can be seen as a multi-threaded tabu search spread throughout the F&F tree with synchronization points at each level of the tree. However, without the use of adaptive memory this represents a simple form of tabu search where memory is limited to the use of a tabu list that may incorporate legitimacy restrictions associated with possible incompatibility among the (elementary) moves used in each path of the tree (i.e. in each compound move).

To complete the examination of the F&F method we shall explain how the algorithm moves from one solution to another when a tree search comes into play, which at the same time justifies the dynamic and adaptive nature of the method. A straightforward explanation is to say that the algorithm stops the tree search and returns to the (single thread) local (or tabu) search procedure once either a new best solution is found (*first improvement* strategy) or when a pre-defined number of levels of the tree have been explored. In the latter, the best trial solution found all over the tree can be used to re-start the search (*best improvement* strategy). In any case, the method is *dynamic* because the number of elementary moves used in the composition of the compound move is not determined in advance, but rather depends on the level in which the "best" trial solution was found, and it is *adaptive* because this level depends on the current state of the search.

III. Filter and Fan for the JSSP

This section describes the basic design of the F&F algorithm implemented for this study. We first point out some general considerations that should be taken into account in the design of a local search algorithm. Then we describe the fundamental components of the proposed algorithm and explain how these components are articulated in the whole process to achieve appropriate levels of performance.

By definition, an algorithm is *robust* if it provides similar quality of solutions over a wide spread of problem instances with significantly different characteristics and sizes. If the quality of these solutions is considered good (e.g. optimal or near-optimal) the algorithm is *effective*. Likewise, the algorithm is *efficient* if these solutions are produced in a relatively short running time. *Simplicity*, is also a very desirable property of a local search algorithm. Simple algorithms are easier to implement, diminish the risk of unexpected errors, and are usually more flexible. By contrast, complex algorithms require longer deployment, are less reliable and usually more difficult to adapt to possibly new problem requirements (or constraints), and so more limited in scope.

In the present study, our primary goal is to design a simple F&F algorithm and evaluate its performance in terms of robustness, effectiveness, and efficiency. As a result, we expect to assess the potential of the F&F approach for the solution of the JSSP and identify needs for improvements on the aforementioned performance measures (or vectors). Note that an attempt to improve one of the performance vectors may not translate into a global improvement over the associated three-dimensional space, as it may deteriorate the performance on some other vectors. Also, the achievement of higher performance levels might require more complex algorithmic designs. However, the investigation of this latter is out of scope of this paper.

The design of a local search algorithm comprises two fundamental components: a procedure that generates a starting solution (needed to initiate the search process) and the definition of a neighborhood structure to locally explore the solution space and generate moves from one solution to another in its neighborhood. Because of the solutions reached depend on the type of the neighborhood structure, complementary neighborhoods or specialized optimization procedures might be necessary in some settings to intensify the search in the neighborhood of high quality solutions as an attempt to find new best solutions. The use of different neighborhoods is tied to the origins of *local search*; however, techniques that are specially conceived to strategically switch to different neighborhoods based on the current stage of the search appeared more recently in association with candidate list strategies in tabu search. Also, the use of supplementary techniques to re-optimize the search at different stages is a classical procedure generally termed *post-optimization*.

We can now undertake the description of the F&F algorithm implemented for this study. The basic structure is as follows.

The simple version of the classical Shifting Bottleneck Procedure (SBP) [2] is used as a constructive algorithm to generate an initial feasible solution. At each step the machine with longest processing time (i.e. the *bottleneck* machine) among the ones that have not been scheduled is selected for scheduling and the method stops when all machines are scheduled. It is well-known that this procedure does not produce high quality solutions per se, but it is convenient to rapidly generate initial feasible solutions for more advance algorithms. A more sophisticated version includes the possibility of backtracking that re-starts the constructive process from scratch using a different machine to initiate the scheduling process. The number of re-starts over a previously scheduled machine is called the *degree of backtracking*. This procedure is also used in our algorithm as a post-optimization technique, as explained later.

The F&F algorithm starts from the solution generated by the SPB and iteratively improves this solution by alternating between the local search and the tree search phases (as explained in the previous section). We consider two types of neighborhoods N_1 (Aarts et al. [1]) and N_2 (Nowichi and Smutnicki [15]) based on classical moves that swap two adjacent operations in the *critical path* (i.e. the longest path in the graph that represents the problem and so the solution). Typically, N_1 swaps arcs that are internal to the blocks of operations in the same machine while N_2 exploit interactions between adjacent blocks by swapping arcs linking operations in different blocks. Depending on the search strategy both types of moves can be used for the local search as well as to define elementary moves in the F&F tree. The use of each type of move and possible combinations of the two will become clearer soon after explaining the proceedings that conduct to different search strategies. At this point we need to formalize the structure of the algorithm and define relevant parameters and components.

The search starts with N_1 neighborhood. Any time, a local optimum is found (in the local search phase) the best $M(0)$ moves (among the M moves evaluated to establish local optimality) are used to create the first level of the F&F neighborhood tree. The next levels are created as follows. Letting η_1 be the number of $M(k)$ moves for level k, the method proceeds by selecting a subset $A_i(k)$ of η_2 moves from $M(0)$ associated with each solution $S_i(k)$ ($i=1,\ldots, \eta_1$) to generate $\eta=\eta_1.\eta_2$ trial solutions for the level $k+1$ (as a result of applying η_2 moves to each solution at level k). Defining $\eta_1=2\eta_2$ has been found a good tradeoff between time complexity and performance of the search. We consider $\eta_1=16$ and $\eta_2=8$. The method stops branching as soon as an improved solution is found, the maximum number of levels L is reached, or if there is no more legitimate candidate moves to evaluate.

In case of a global improvement is found in the tree search the new best solution is made the starting solution for another run of the local search procedure. However, if the solution at the root node could not be improved, the method switches back to the local search starting with the best trial solution encountered in the tree search and using neighborhood N_2. In this case, the list M determined in the last run of the local search procedure, and so made up of type N_1 moves, is now extended with new candidates of type N_2. The new list

M(0) is created using the best moves of each type in equal number. The objective is to allow the algorithm to combine both types of neighborhoods throughout the F&F tree.

The importance of the use of an alternative neighborhood is reinforced by the following observation. The process of selecting η_2 moves has to obey a set a legitimacy conditions specific to the type of move utilized and that are established to identify possible incompatibility between elementary moves. Thus, legitimacy conditions generate legitimacy restrictions that prevent incompatible moves from being combined. For example, both N_1 and N_2 neighborhoods operate on adjacent job operations. If a move at a given level of the tree destroyed the adjacency of two operations, then all the moves associated with the swap of these two operations must be excluded from the candidates for the corresponding path of the tree. For this reason, it is possible that for some solutions the number of legitimate moves for a specific neighborhood is too restricted to allow for an improvement, and therefore the utilization of an alternative neighborhood permits to enlarge the neighborhood space and so increases the chances to find better solutions.

IV. Experimental Results

To measure the performance of the proposed algorithm a set of tests were carried out on four classes of benchmark problems created by the authors shown in Table I.

Table I – Benchmark problems

Class	Author(s)	Nr. of problems
LA	Lawrence [12]	40
FT	Fisher and Thompson [6]	3
ABZ	Adams, Balas and Zawack [2]	5
ORB	Applegate and Cook [3]	10

The problems size varies between 6 and 30 tasks and between 5 and 15 machines. It is well known that some problems are more difficult than others, such as LA6-10, LA11-15, LA30-35, which chiefly correspond to cases where the number of jobs is far superior to the number of machines. Usually, JSSP problems become harder to solve as the problem size increases or the number of jobs gets closer to the number of machines. As stated in Vaessens [22] the hardest instances are considered to be: FT10, LA2, LA19, LA21, LA24, LA25, LA27, LA29, LA36-40.

Tables II to V summarize, for each class of problems, the results obtained by the F&F algorithm. In each table, the first three columns provide the problem name and size in terms of the number of jobs (N) and machines (M). Column OPT reports the optimum solution or the best known (for instances ABZ8, ABZ9 and LA29). The next columns report the solution values produced by the F&F algorithm and the corresponding percentage deviation relatively to OPT values. The last column reports the running times (in seconds) necessary for the algorithm to find its best solution. (All the runs were carried out on a 1.7 GHz Pentium IV 256MB of RAM computer, the algorithm was coded in C language and compiled with MS Visual C++ 6.0 under a Windows 2000 Professional platform.)

Table II – F&F results for LA problems

Problem	Jobs	Machines	OPT	F&F	%	CPU Time
LA 1	10	5	666	666	**0.000**	1
LA 2	10	5	655	655	**0.000**	2
LA 3	10	5	597	597	**0.000**	1
LA 4	10	5	590	590	**0,000**	1
LA 5	10	5	593	593	**0.000**	1
LA 6	15	5	926	926	**0.000**	1
LA 7	15	5	890	890	**0.000**	1
LA 8	15	5	863	863	**0.000**	1
LA 9	15	5	951	951	**0.000**	1
LA 10	15	5	958	958	**0.000**	1
LA 11	20	5	1222	1222	**0.000**	1
LA 12	20	5	1039	1039	**0.000**	1
LA 13	20	5	1150	1150	**0.000**	1
LA 14	20	5	1292	1292	**0.000**	1
LA 15	20	5	1207	1207	**0.000**	1
LA 16	10	10	945	947	0.212	1
LA 17	10	10	784	784	**0.000**	2
LA 18	10	10	848	848	**0.000**	1
LA 19	10	10	842	846	0.475	1
LA 20	10	10	902	907	0.554	3
LA 21	15	10	1046	1052	0.574	4
LA 22	15	10	927	927	**0.000**	11
LA 23	15	10	1032	1032	**0.000**	1
LA 24	15	10	935	941	0.642	15
LA 25	15	10	977	982	0.512	27
LA 26	20	10	1218	1218	**0.000**	3
LA 27	20	10	1235	1242	0.567	10
LA 28	20	10	1216	1225	0.740	5
LA 29	20	10	1142	1176	2.977	36
LA 30	20	10	1355	1355	**0.000**	1
LA 31	30	10	1784	1784	**0.000**	1
LA 32	30	10	1850	1850	**0.000**	1
LA 33	30	10	1719	1719	**0.000**	1
LA 34	30	10	1721	1721	**0.000**	1
LA 35	30	10	1888	1888	**0.000**	1
LA 36	15	15	1268	1281	1.025	4
LA 37	15	15	1397	1418	1.503	4
LA 38	15	15	1196	1213	1.421	16
LA 39	15	15	1233	1250	1.379	19
LA 40	15	15	1222	1228	0.491	44
Average					*0.327*	*5.7*

Table III – F&F results for FT problems

Problem	Jobs	Machines	OPT	F&F	%	CPU Time
FT 06	6	6	55	55	**0.000**	1
FT 10	10	10	930	938	0.860	13
FT 20	20	5	1165	1165	**0.000**	10
Average					*0.287*	*8.0*

Table IV – F&F results for ABZ problems

Problem	Jobs	Machines	OPT	F&F	%	CPU Time
ABZ 5	10	10	1234	1236	0.162	13
ABZ 6	10	10	943	943	**0.000**	1
ABZ 7	20	15	656	669	1.982	47
ABZ 8	20	15	665	687	3.308	51
ABZ 9	20	15	679	706	3.976	77
Average					*1.886*	*37.8*

Table V – F&F results for ORB problems

Problem	Jobs	Ma-chines	OPT	F&F	%	CPU Time
ORB 1	10	10	1059	1064	0.472	7
ORB 2	10	10	888	897	1.014	1
ORB 3	10	10	1005	1022	1.692	15
ORB 4	10	10	1005	1024	1.891	2
ORB 5	10	10	887	894	0.789	3
ORB 6	10	10	1010	1012	0.198	14
ORB 7	10	10	397	397	**0.000**	11
ORB 8	10	10	899	899	**0.000**	6
ORB 9	10	10	934	944	1.071	1
ORB 10	10	10	944	944	**0.000**	5
Average					*0.713*	*6.5*

The results show that the F&F algorithm finds the optimal solution for 32 out of 58 problems, which represents more than 55% of the problems tested. Also, the algorithm produces solutions that are on average at 1.89, 0.29, 0.33, and 0.71 percent above the optimum (or best known) solutions for classes ABZ, FT, LA and ORB, respectively. This results in an overall percentage deviation of 0.80% on average. As far as the computational time is concerned we can see that the algorithm finds its best solutions in relatively short time: less than 40 seconds on average for ABZ problems and no more than 8 seconds on average for classes FT, LA and ORB. In addition, it should be noticed that 1 second of running time was enough for the algorithm to find the optimal solution for 30 out of the 38 instances that the optimal solution was found.

The quality of the solutions and the associated running times reported in these results clearly indicate that the proposed algorithm is very effective and efficient in solving JSSP.

V. Conclusions

Effective neighborhood constructions that generate compound moves have been the source of several recent developments in local search. This paper proposed a new algorithm for the JSSP based on a Filter and Fan (F&F) approach. F&F is an iterative local search method that explores the solution space by generating compound moves in a tree search fashion. The effectiveness of the method underlies the creation and maintenance of two candidate lists—the *filter* and the *fan* candidate lists—used to generate complex moves with reasonable computational effort. Although the purpose of this study was to provide a first but preliminary study on the application of F&F to the JSSP, it turned out that even a simplistic design of the method led to quite impressive results in both solution quality and computation time. However, the fact that the algorithm did not find the optimal solution for some of the hardest problems tested, suggests the continuation of this research toward the development of more advanced forms of the F&F method. To this end, further investigation should include the consideration of candidate lists of neighborhoods that are not limited to adjacent operations or restricted to the critical path. In addition, tabu search may be of great help to diversify the search and in identifying potential good moves determined by the use adaptive memory.

References

[1] E.H.L. Aarts, P.J.M. Van Laarhoven, J.K. Lenstra and N.L.J. Ulder, “A Computational Study of Local Search Algorithms for Job Shop Scheduling”, ORSA Journal on Computing, 6(2), 1994, pp 118-125.

[2] J. Adams, E. Balas and D. Zawack, “The Shifting Bottleneck Procedure for Job Shop Scheduling”, Management Science, 34 (3), 1988, pp. 391-401.

[3] D. Applegate and W. Cook, “A Computational Study of the Job-Shop Scheduling Problem”, ORSA Journal on Computing, 3 (2), 1991, pp. 149-156.

[4] E. Balas and A. Vazacopoulos, “Guided Local Search with Shifting Bottleneck for Job Shop Scheduling”, Management Science, 44 (2), 1998, pp. 262-275.

[5] L. Cavique, C. Rego and I. Themido, "Subgraph Ejection Chains and Tabu Search for the Crew Scheduling Problem", Journal of the Operational Research Society, 50, 1999, pp. 608-616.

[6] H. Fisher and G. L. Thompson, “Probabilistic Learning Combinations of Local Job Shop Scheduling Rules”, in J. F. Muth and G. L. Thompson (Eds.), *Industrial Scheduling*, Prentice Hall, Englewood Cliffs, New Jersey, USA, 1963, pp. 225-251.

[7] D. Gamboa, C. Rego and F. Glover, “Data Structures and Ejection Chains for Solving Large Scale Traveling Salesman Problems”, European Journal of Operational Research, to appear.

[8] D. Gamboa, C. Rego and F. Glover, “Implementation Analysis of Efficient Heuristic Algorithms for the Traveling Salesman Problem”, Computers and Operations Research, to appear.

[9] F. Glover, “A Template for Scatter Search and Path Relinking”, in J. K. Hao, E. Lutton, E, Ronald, M. Schoenauer and D. Snyers (Eds.), *Artificial Evolution*, Lecture Notes in Computer Science, Springer, Heidelberg, vol. 1363, 1998, pp. 13-54.

[10]J. Grabowski and M. Wodecki, “A Very Fast Tabu Search Algorithm for the Job Shop Problem”, in C. Rego and B. Alidaee (Eds.), *Metaheuristic Optimization via Memory and Evolution: Tabu Search and Scatter Search*, Kluwer Academic Publishers, 2004.

[11]P. Greistorfer, C. Rego and B. Alidaee, “A Simple Filter and Fan Approach to the Facility Location Problem”, Research Report HCES-02-04, Hearin Center for Enterprise Science, School of Business Administration, University of Mississippi, USA, 2004.

[12]S. Lawrence, “Supplement to Resource Constrained Project Scheduling: An Experimental Investigation of Heuristic Scheduling Techniques”, Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, USA, 1984.

[13]S. Lin and B. Kernighan, “An Effective Heuristic Algorithm for the Traveling Salesman Problem”, Operations Research, 21, 1973, pp. 498-516.

[14]B.T. Lowerre, “The HARPY Speech Recognition System”, Ph.d. thesis, Carnegie-Mellon University, USA, April 1976.

[15]E. Nowichi and C. Smutnicki, “A Fast Taboo Search Algorithm for the Job Shop Problem”, Management Science, 42(6), 1996, pp 797-813.

[16]E. Nowichi and C. Smutnicki, “Some New Ideas in Tabu Search for Job Shop Scheduling”, in C. Rego and B. Alidaee (Eds.), *Metaheuristic Optimization via Memory and Evolution: Tabu Search and Scatter Search*, Kluwer Academic Publishers, 2004.

[17]C. Rego, “Relaxed Tours and Path Ejection Chains for the Traveling Salesman Problem”, European Journal of Operational Research, 106, 1998, pp. 522-538.

[18]C. Rego, “A Subpath Ejection Method for the Vehicle Routing Problem”, Management Science, 44, 1998, pp. 1447-1459.

[19]C. Rego, “Node Ejection Chains for the Vehicle Routing Problem: Sequential and Parallel Algorithms”, Parallel Computing, 27, 2001, pp. 201-222.

[20]C. Rego and F. Glover, “Local Search and Metaheuristics for the Traveling Salesman Problem”, in G. Gutin and A. Punnen (Eds.), *The Traveling Salesman Problem and its Variations*, Kluwer Academic Publishers, Combinatorial Optimization Series, 12, 2002, pp. 309-368.

[21]I. Sabuncuoglu and M. Bayiz, “Job shop Scheduling with Beam Search”, European Journal of Operational Research 118, 1999, pp. 390-412.

[22]R. J.M. Vaessens, E. H. L. Aarts and J. K. Lenstra, “Job Shop Scheduling by Local Search”, INFORMS Journal on Computing, 8 (3), 1996, pp. 302-317.

[23]M. Yagiura, T. Ibaraki and F. Glover, “An Ejection Chain Approach for the Generalized Assignment Problem”, INFORMS Journal on Computing, 16, 2004, to appear.