AU J.T. 10(1): 11-22 (Apr. 2006)

Flow-shop Problems: A Heuristic Search Algorithm

Graham Winley

Faculty of Science and Technology, Assumption University
Bangkok, Thailand

Abstract

This article presents an intelligent heuristic search algorithm (IHSA) for solving flow-shop problems with two or three machines and an arbitrary number of jobs. Following its initial development the algorithm has been modified in two different ways in order to reduce backtracking and to improve its performance. The first modification concerns the choice of the best heuristic function to use, and the second modification concerns the way in which heuristic estimates at nodes on the search path are determined as the search progresses. Experimental evidence of the improved performance of the algorithm as a result of each of these modifications is presented.*

Keywords: *Admissible heuristic function/estimate, dominance, flow- shop problems, heuristic search algorithm*

Introduction

The solution to the general flow-shop scheduling problem involving *n* jobs and *m* machines determines the sequence of jobs on each machine in order to complete all the jobs on all the machines in the minimum total time (i.e. with minimum makespan) where each job is processed first on machine *1* then on machine *2* and so on until it is finally processed on machine *m*. The complexity of the general problem is of the order $(n!)^m$. It is known that there is an optimal solution where the sequence of jobs is the same on the first two machines and the sequence of jobs is the same on the last two machines (Conway *et al*. 1967). Consequently, for a flow-shop problem with two or three machines there is an optimal solution where the jobs are processed in the same sequence on each machine and in this case it is among *n!* possible sequences.

Early research on flow-shop problems is based mainly on Johnson's theorem, which gives a procedure for finding an optimal solution with only two machines, or three machines with certain characteristics (Johnson 1954; Kamburowski 1997). Other approaches for the general problem include integer linear programming and combinatorial programming, which use intensive computation to obtain optimal solutions and are generally not feasible from a computational standpoint because the number of variables increases exponentially as the number of machines increases (Pan 1997). The branch-and-bound method uses upper or lower bounds to guide the direction of the search. Depending on the effectiveness of the heuristic and the search strategy this method may return only near optimal solutions but with long computation time (Ignall and Schrage 1965; Lomnicki 1965; McMahon and Burton 1967; Pan and Chen 1997). The genetic algorithm has been applied to these problems and, depending on the nature of the problem, it may find an optimal solution but, due to the evolutionary nature of this approach, the computation time is unpredictable (Cleveland and Smith 1989; Chen *et al*. 1996). Also, heuristic search methods and methods based on fuzzy logic have been applied to the general flow-shop problem (Lai 1996; Wang *et al*. 1996; Hong and Wang 1999; Hong *et al*. 2000; Hong and Chuang 1998a, 1998b, 1999; Hong *et al*. 1998). In particular, Fan (1999a, 1999b, 2002) developed an intelligent heuristic search algorithm ($IHSA^*$) for 2 or 3 machine problems based on the Search and Learning A^* (SLA^*) algorithm (Zamani and Shue 1995, 1998; Zamani 2001), which is a modified and improved version of the Learning Real Time

A^* ($LRTA^*$) algorithm (Korf 1990, 1993), which is, in turn, an improved version of the original A^* algorithm (Hart *et al.* 1968; Gheoweth and Davis 1991).

At the start of the search using $IHSA^*$ it is assumed in turn that each of the n jobs is placed first in the job sequence and in each case an estimate is made of the total time needed to complete all the jobs on all the machines. Among these n heuristic estimates the smallest estimate is selected as the value of the heuristic function and the search begins by placing the corresponding job first in the job sequence. If the heuristic function value does not exceed the minimum makespan then the heuristic function is admissible and $IHSA^*$ is complete and optimal. The proof of this result for $IHSA^*$ by Fan (2002) is similar to that by Korf (1985a, 1985b) in relation to the A^* algorithm.

The purpose of this article is to present and illustrate a modified version of $IHSA^*$, which incorporates two modifications that have been made to it since its initial development by Fan (1999a, 1999b, 2002). The first modification determines the best admissible heuristic function to use for a given problem. The second modification determines the procedure to use to calculate heuristic estimates at nodes on the search path as the search progresses. Each of these modifications is discussed and proofs are given for results related to these modifications. For each of the modifications experimental evidence is presented to illustrate the improvement in the performance of the algorithm.

The Initial Version of $IHSA^*$

Table 1 presents the known information for a flow-shop problem involving *n* jobs and three machines. The times required to process each job on each machine (a_i, b_i, c_i) are assumed to be positive integers.

Fig.1 illustrates the processing of jobs in the sequence ϕ_{st} on three machines M_1, M_2, M_3, where $\phi_{st} = \{J_s, \ldots, J_t\}$ is a sequence of the *n* jobs with J_s scheduled first and J_t scheduled last. $T(\phi_{st})$ is the makespan for the job sequence ϕ_{st}, and $S(\phi_{st})$ is the time at which all jobs in ϕ_{st} are completed on machine M_2.

Table 1. Flow-shop problems
(*n* jobs, *3* machines)

Jobs/Machines	M_1	M_2	M_3
J_1	a_1	b_1	c_1
J_2	a_2	b_2	c_2
J_3	a_3	b_3	c_3
..	..	..	..
J_n	a_n	b_n	c_n

From Fig. 1 it is seen that,

$$H_1 = \min [a_1 + b_1, a_2 + b_2, \ldots, a_n + b_n] + \sum c \quad (1)$$

is an admissible heuristic function as it will underestimate the minimum time taken to complete all of the jobs on all of the machines and this is the heuristic function that was used in the initial version of the algorithm. Other admissible heuristic functions may be developed and the method for selecting the best among all of these is discussed below in relation to the first modification to the initial version of $IHSA^*$.

Before presenting the initial version of the algorithm the underlying state transition process is described together with the characteristics of search path diagrams which are used to illustrate the progress of the search when the algorithm is implemented.

The State Transition Process

The operation done by machine *j* on job *i* is represented by M_jJ_i and has a duration given by one of the values a_i, b_i, c_i. At any time during the scheduling process each of the 3n operations will be in only one of three states, namely the "finished" state, the "in-progress" state, and the "not scheduled" state. At any time the state level of the scheduling process is the number of operations in the "finished" state and a state transition occurs when one or more of the operations move from the "in-progress" state to the "finished" state. The algorithm describes the procedure which takes the search process from one state level to the next and the development of the search path is illustrated graphically using search path diagrams.

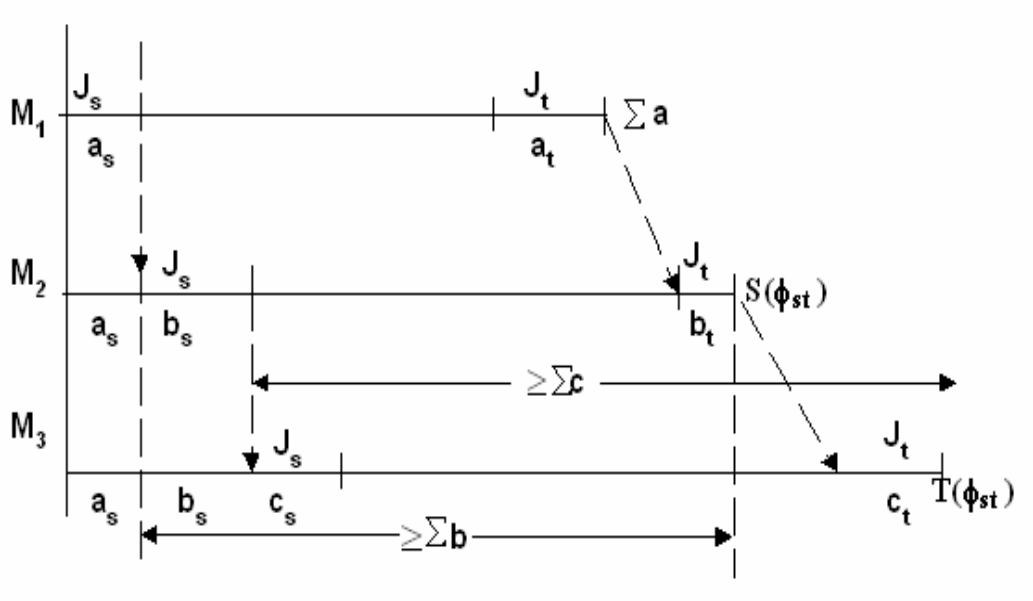


Fig. 1: Processing the job sequence ϕ_{st}

Search Path Diagrams

A search path diagram consists of nodes drawn at each state level L with one of the nodes connected by arcs to nodes at the next state level L + 1. Each node contains as many cells as there are machines and cells 1, 2, 3 are used to display information about possible operations on machines M_1, M_2, M_3, respectively. When a state transition occurs one of the nodes at the current state level L is expanded and it is connected to nodes at the next state level L + 1 where each node represents exactly one of the different possible ways of starting operations in the "not scheduled" state on the available machines. At each of these nodes cell *j* is labeled with a job J_i, which is either in progress on M_j or is one of the jobs that may start on M_j, as well tM_jJ_i which is the remaining time needed to complete J_i on M_j. A blank cell indicates that the machine is idle.

Near each node the heuristic estimate (h) associated with the node is recorded. The heuristic estimate (h) is determined by using the heuristic function chosen at the start of the search in conjunction with a procedure which is used in Step 2 of the algorithm. It is an estimate of the time required to complete an operation at the node identified by the procedure as well as all the operations that are not in the "finished" state. At each state level the node selected for expansion is the one which has associated with it the minimum heuristic estimate among all the estimates at the nodes at that state level. Near this node the value of f = h + k is recorded where the edge cost (k) is the time that has elapsed since the preceding state transition occurred.

A comparison between f and h' is made where h' is the minimum heuristic estimate at the preceding state level. Based on that comparison the search path either backtracks to the preceding state level or moves forward to the next state level. If backtracking occurs the value of the heuristic estimate h' at the preceding state level is updated to the value of f and the search moves back to that previously expanded node. If the path moves forward then below the expanded node the edge cost (k) is recorded and it will be the smallest value among all of the tM_jJ_i values in the other cells at the expanded node. For convenience of presentation a new search path diagram is drawn when backtracking in the previous diagram is completed.

At state level 0 there are n (root) nodes corresponding to the number of jobs and there will be information recorded in only the first cell because no operation can start on any machine other than M_1. The final search path diagram represents the optimal solution and traces a path from one of the root nodes to a terminal node where h = f = 0. At that root

node the value of h, or f, is the minimum makespan for the problem and the optimal job sequence can be read sequentially by recording the completed operations down the search path from the root node to the terminal node.

IHSA* (Initial Version)

Step1: At state level 0 expand the node identified by calculating the value of H_1, from (1), and move to the nodes at state level 1.

Step 2: At the current state level if the heuristic estimate of one of the nodes has been updated by backtracking use the updated value as the heuristic estimate for that node and proceed to Step 3. Otherwise, calculate a heuristic estimate for each node using Procedure 1, which is described below.

Step3: Select the node with the smallest heuristic estimate among all those calculated in Step 2. If it is necessary then break ties randomly.

Step 4: Calculate $f = h + k$ where h is the heuristic estimate found in Step 3 and k (edge cost) is the time that has elapsed since the preceding state transition occurred.

Step 5: If $f > h'$, where h' is the minimum heuristic estimate calculated at the preceding state level, then backtrack to that preceding state level and increase the value of h' at that preceding node to the current value of f and repeat Step 4 at that node.

Step 6: If $f \leq h'$ then proceed to the next state level and repeat from Step 2.

Step 7: If $f = 0$ and $h = 0$ then Stop.

In Procedure 1 the heuristic estimate for a node is based on an operation in only one of the cells at the node and operations in the other two cells are not taken into account. For example, if cell 1 is not blank then the estimate is based on the operation in cell 1 and operations in the other two cells are not considered. If cell 1 is blank then the estimate is based on the operation in cell 2, and cell 3 is ignored. An operation in cell 3 is only considered if the other cells are blank.

Modifications to IHSA*

A Modification to Step 1 of the Initial Version of IHSA*

Following the initial development of IHSA* Winley and Fan (2006a) developed and compared nine admissible heuristic functions which may be used in conjunction with the algorithm. Among these nine functions three are identified as having initial values which are closer to the minimum makespan than the other six and choosing the function among these

Procedure 1: For a node at the current state level,

(a) If cell 1 is labelled with M_1J_i then the heuristic estimate h for the node is based on the operation in cell1 and is given by,

$$h = \begin{cases} a_i + b_i + C_1; \text{ for } M_1J_i \text{ in the "not scheduled" state,} \\ t\ M_1J_i + b_i + c_i + C_1; \text{ for } M_1J_i \text{ in the "in-progress" state,} \end{cases} \quad (2)$$

where, $C_1 = \sum_j c_j$ for all j such that M_1J_j is in the "not scheduled" state.

(b) If cell 1 is blank, and cell 2 is labelled with M_2J_i then the heuristic estimate h for the node is based on the operation in cell 2 and is given by,

$$h = \begin{cases} b_i + C_2; \text{ for } M_2J_i \text{ in the "not scheduled" state,} \\ t\ M_2J_i + c_i + C_2; \text{ for } M_2J_i \text{ in the "in-progress" state,} \end{cases} \quad (3)$$

where, $C_2 = \sum_j c_j$ for all j such that M_2J_j is in the "not scheduled" state.

(c) If cell 1 and cell 2 are blank, and cell 3 is labelled with M_3J_i then the heuristic estimate h for the node is based on the operation in cell 3 and is given by,

$$h = \begin{cases} C_3; \text{ for } M_3J_i \text{ in the "not scheduled" state,} \\ t\ M_3J_i + C_3; \text{ for } M_3J_i \text{ in the "in-progress" state,} \end{cases} \quad (4)$$

where, $C_3 = \sum_j c_j$ for all j such that M_3J_j is in the "not scheduled" state.

three with the largest initial value as the one to three use in Step 1 of the algorithm improves the performance of the algorithm in terms of the number of: nodes expanded; backtracks required; and algorithm steps executed. Also, in the case where one machine dominates the others the best heuristic function among the functions can be determined without the need to compute the value of each of these three functions. The derivation of these functions, the proofs of their admissibility, and the case where one machine dominates the others, are presented in the Appendix.

Table 2: Performance of IHSA* using the modification to Step 1

Problem	Heuristic Function Used	Value of Heuristic Function	Performance Characteristics			Minimum Makespan (Optimal Sequence)
			Number of Nodes Expanded	Number of Backtracks	Number of Algorithm Steps	
1	H_3 H_1 H_2	17 10 9	13 22 21	0 11 12	10 33 34	17 (J_1, J_2, J_3)
2	H_3 H_2 H_1	25 17 15	13 22 25	0 11 14	10 32 38	25 (J_2, J_3, J_1)
3	H_2 H_1 H_3	24 15 13	13 36 36	6 39 37	22 88 84	24 (J_1, J_3, J_2)
4	H_2 H_3 H_1	24 15 13	17 38 38	10 41 42	30 92 94	24 (J_2, J_3, J_1)
5	H_1 H_2 H_3	28 20 18	13 17 17	0 3 3	10 16 16	28 (J_2, J_1, J_3)
6	H_1 H_3 H_2	26 19 16	14 18 18	0 3 3	11 17 17	26 (J_1, J_2, J_3)

Table 3: Performance of IHSA* with Step 1 modified and a dominant machine

Problem (Dominant Machine)	Heuristic Function Used	Value of Heuristic Function	Performance Characteristics			Minimum Makespan (Optimal Sequence)
			Number of Nodes Expanded	Number of Backtracks	Number of Algorithm Steps	
1 (M_1)	H_3 H_1 H_2	29 19 17	14 28 31	0 20 25	11 51 61	29 (J_1, J_3, J_2)
2 (M_2)	H_2 H_3 H_1	29 20 19	19 41 41	12 48 51	35 107 113	29 (J_2, J_3, J_1)
3 (M_3)	H_1 H_2 H_3	29 23 19	12 16 16	0 3 3	9 15 15	29 (J_2, J_1, J_3)

From the results in the Appendix the three best functions to consider in Step 1 of the algorithm are:

$$\left.\begin{aligned} H_1 &= \min[a_1 + b_1, a_2 + b_2, \ldots \\ &\qquad , a_n + b_n] + \sum c, \\ H_2 &= \min[a_1 + u_1, a_2 + u_2, \ldots \\ &\qquad , a_n + u_n] + \sum b, \end{aligned}\right\} \quad (5)$$

$$H_3 = \min[v_1, v_2, \ldots, v_n] + \sum a,$$

where, $u_k = \min[c_1, c_2, \ldots, c_{k-1}, c_{k+1}, \ldots, c_n]$, and $v_k = \min[b_1 + c_1, b_2 + c_2, \ldots, b_{k-1} + c_{k-1}, b_{k+1} + c_{k+1}, \ldots, b_n + c_n]$.

Consequently, Step 1 of the algorithm becomes:

Step1: At state level 0 expand the node identified by calculating max[H_1, H_2, H_3], from (5), and move to the nodes at state level 1.

Table 2 shows experimental evidence from 6 problems where using this modification to Step 1 in IHSA* results in an improvement in its performance through a reduction in the number of: nodes expanded; backtracking steps; and algorithm steps executed.

Table 3 shows 3 problems where there is a dominant machine. Again, this experimental evidence shows that using the modification to Step 1 in IHSA* improves its performance.

A Modification to Step 2 of the Initial Version of IHSA

The second modification to IHSA*, proposed by Winley and Fan (2006b), is based on the principle that when heuristic estimates for all the nodes at a state level are being calculated it is desirable to obtain the largest possible heuristic estimate at each node before selecting the node with the smallest admissible estimate as the node to be expanded. The larger the value of this admissible estimate the less likely it is that the search will need to backtrack and this will result in an improvement in the performance of the algorithm. In the initial version of IHSA* the use of Procedure 1 in Step 2 gives a heuristic estimate for a node that is based on considering an operation in only one of the cells at the node while operations in the other two cells are not taken into account.

The second modification affects Procedure 1 and involves calculating a heuristic estimate for each cell at a node and then using the largest of these estimates at the cells as the heuristic estimate for the node. This is done for each node at the current state level and then, as before, the minimum heuristic estimate among the nodes is admissible and identifies the node to be expanded.

Procedure 2: For a node at the current state level,

(a) For cell 1:

If the cell is blank then $h_1 = 0$. Otherwise, h_1 is given by (2).

(b) For cell 2:

If the cell is blank then $h_2 = 0$. Otherwise, h_2 is given by (3).

(c) For cell 3:

If the cell is blank then $h_3 = 0$. Otherwise, h_3 is given by (4).

This version incorporates the heuristic function H_1. It is easily changed to incorporate H_2 or H_3 for problems where these functions have been selected in Step 1 of the algorithm.

Only Step 2 in the initial version of IHSA* needs to be modified so that the new Step 2 becomes:

Step 2: At the current state level if the heuristic estimate of one of the nodes has been updated by backtracking use the updated value as the heuristic estimate for that node and proceed to Step 3. Otherwise, at each node use Procedure 2 to calculate h_1, h_2, h_3 and use max (h_1, h_2, h_3) as the heuristic estimate for the node.

The use of Procedure 2 in Step 2 of the algorithm will produce the same heuristic estimate as the use of Procedure 1 if and only if one of the following 3 conditions is satisfied: $h_1 = \max(h_1, h_2, h_3)$; $h_2 = \max(h_1, h_2, h_3)$ and cell 1 is blank; or $h_3 = \max(h_1, h_2, h_3)$ and cell 1, and cell 2 are blank. Under any other conditions Procedure 2 will produce a heuristic estimate at a node which is larger than the estimate given by Procedure 1 and in practical problems this has been observed to occur often enough to recommend the use of Procedure 2.

Table 4 shows experimental evidence from 6 problems where the performance of

IHSA* using only the modification to Step 1 is compared to the performance of the algorithm using both of the modifications to Steps 1 and 2. It is evident that using both of the modifications further improves the performance characteristics of the algorithm.

The Modified Version of IHSA*

The experimental evidence from Tables 2, 3 and 4 indicates that the modifications to Steps 1 and 2 of the initial version of IHSA* should be incorporated into a modified version of the algorithm which is:

Step1: At state level 0 expand the node identified by calculating max[H_1, H_2, H_3], from (5), and move to the nodes at state level 1.

Step 2: At the current state level if the heuristic estimate of one of the nodes has been updated by backtracking use the updated value as the heuristic estimate for that node and proceed to Step 3. Otherwise, at each node use Procedure 2 to calculate h_1, h_2, h_3 and use max (h_1, h_2, h_3) as the heuristic estimate for the node.

Table 4: Performance of IHSA*: comparing the use of the modification to Step1 with the use of the modifications to Step 1 and Step 2

Problem	Best Heuristic Function	Steps Modified in the Initial Version of IHSA*	Performance Characteristics			Minimum Makespan (Optimal Sequence)
			Number of Nodes Expanded	Number of Backtracks	Number of Steps	
1	H_3	1	21	12	34	16
		1 and 2	15	6	22	(J_2, J_1, J_3)
2	H_3	1	43	36	81	21
		1 and 2	28	16	41	(J_3, J_2, J_1)
3	H_2	1	32	23	56	18
		1 and 2	19	5	20	(J_1, J_2, J_3)
4	H_2	1	28	20	49	21
		1 and 2	16	3	15	(J_2, J_1, J_3)
5	H_1	1	20	17	44	28
		1 and 2	13	0	10	(J_2, J_1, J_3)
6	H_1	1	21	16	43	26
		1 and 2	14	0	11	(J_1, J_2, J_3)

Step3: Select the node with the smallest heuristic estimate among all those calculated in Step 2. If it is necessary then break ties randomly.

Step 4: Calculate $f = h + k$ where h is the heuristic estimate found in Step 3 and k (edge cost) is the time that has elapsed since the preceding state transition occurred.

Step 5: If $f > h'$, where h' is the minimum heuristic estimate calculated at the preceding state level, then backtrack to that preceding state level and increase the value of h' at that preceding node to the current value of f and repeat Step 4 at that node.

Step 6: If $f \leq h'$ then proceed to the next state level and repeat from Step 2.

Step 7: If $f = 0$ and $h = 0$ then Stop.

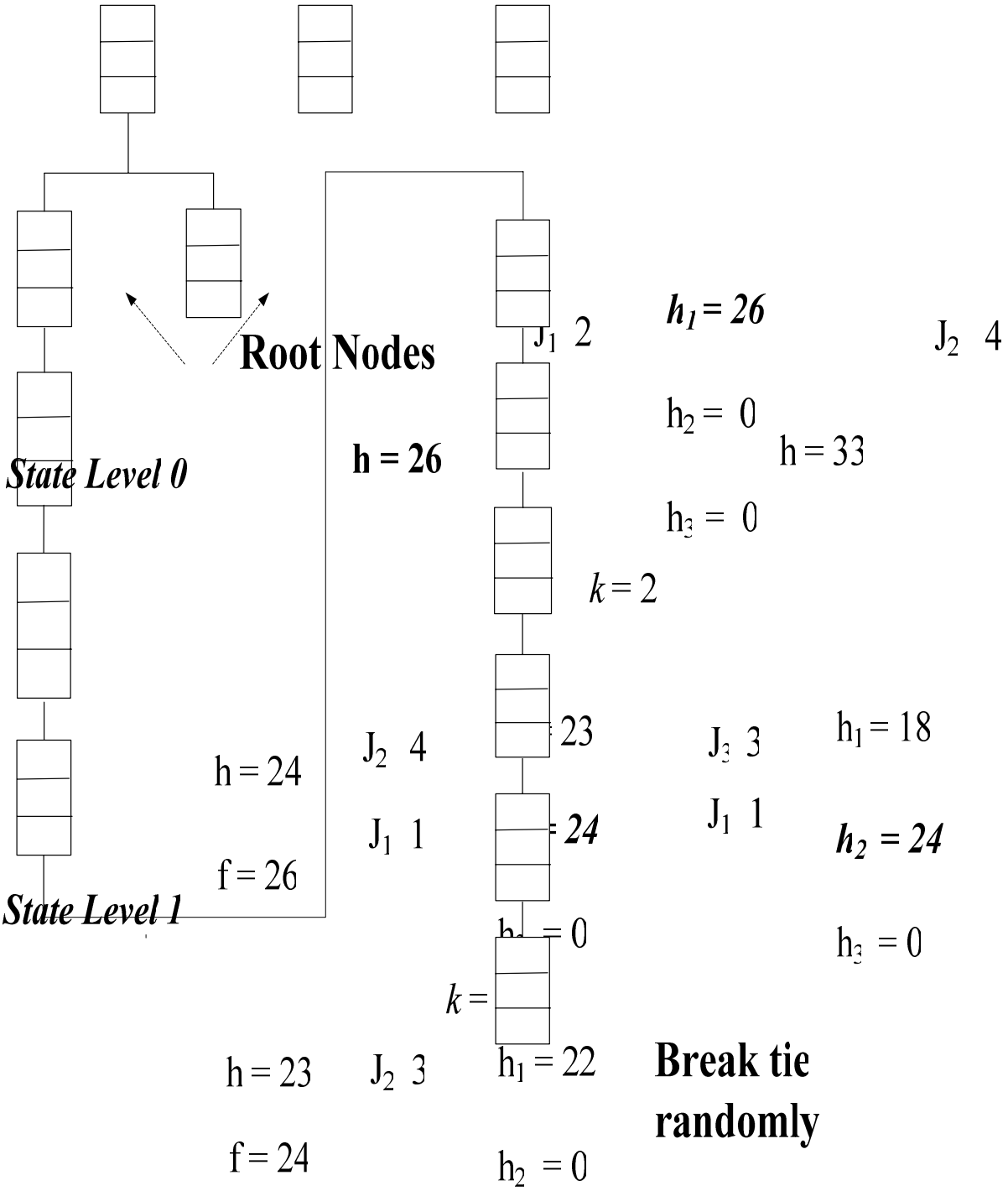
An Example

The problem defined in Table 5 is used to illustrate the use of the modified version of IHSA*.

For this problem, from (5), $H_1 = 26$, $H_2 = 16$, $H_3 = 19$, and so H_1 is used in Step 1 of the algorithm. The solution of the problem requires only 1 search path diagram, as shown in Fig. 2.

Table 5: Flow-shop problem

Jobs/Machines	M_1	M_2	M_3
J_1	2	1	10
J_2	4	6	5
J_3	3	2	8



Conclusion

The development of an intelligent heuristic search algorithm (IHSA*) for solving flow-shop scheduling problems involving 2 or 3 machines and an arbitrary number of jobs has been described. Starting with the initial version of IHSA* 2 modifications are presented.

The first modification relates to Step 1 of the algorithm and concerns the choice of an admissible heuristic function which is as close as possible to the minimum makespan for the given problem. A set of 9 possible functions is

derived, their admissibility is proved, and it is shown that the best function to use in Step 1 of the algorithm is the function with the largest value among a subset of 3 (H_1, H_2, H_3) of these nine functions. In the particular case where one machine dominates the others it is shown that the best function can be identified without the need to calculate the value of each of the three functions.

The second modification concerns the procedure used in Step 2 of the algorithm to determine heuristic estimates at nodes on the search path as the search progresses. The initial version of the algorithm determines a heuristic estimate at a node by considering an operation on only one of the machines, each of which is identified in one of the cells at a node. The modification to this procedure determines a heuristic estimate at a node by calculating an estimate for each of the three operations at a node and then selecting the largest among these estimates as the heuristic estimate for the node. This modified procedure will never produce an estimate that is less than the estimate produced by the procedure used in the initial version of the algorithm, and in many problems it will be larger. When choosing the next node to expand, all of the nodes which may be expanded are considered, and the one with the smallest heuristic estimate is selected. Consequently, in many problems the modification will result in selecting a node for expansion which has a larger heuristic estimate than the estimate determined by using the procedure in the initial version of the algorithm. This larger estimate, which is also admissible, reduces the chance that the search will need to backtrack and this improves the performance of the algorithm.

Three sets of experimental results are presented to illustrate the improvements in the performance characteristics of the algorithm which result from incorporating the modifications. The performance characteristics are measured by the number nodes expanded; backtracking steps; and algorithm steps executed.

The first two sets of results illustrate improvements in the performance characteristics which result from modifying only Step 1 of the algorithm and, in particular, the second set of results involves problems where one of the machines dominates the others. The third set of results illustrates the further improvement in performance gained by incorporating the modifications to both Step 1 and Step 2.

An example is presented using a search path diagram to illustrate the operation of the modified version of the IHSA*, which incorporates the modifications to Step 1 and Step 2.

References

Chen, C.L; Neppalli, R.V.; and Aljaber, N. 1996. Genetic algorithms applied to the continuous flow shop problem. Comp. Ind. Engineer. 30: 919-29.

Cleveland, G.A.; and Smith, S.F. 1989. Using genetic algorithms to schedule flow shop. Proc. 3rd Conf. Genetic Algorithms, pp. 160-9.

Conway, R.W.; Maxwell, W.L.; and Miller, L.W. 1967. Theory of Scheduling. Addison-Wesley, Reading, Massachusetts, USA. .

Fan, J.P.O. 1999a. The development of a heuristic search strategy for solving the flow-shop scheduling problem. Proc. IASTED Int. Conf. Applied Informatics, Innsbruck, Austria, pp. 516-8.

Fan, J.P.O.1999b. An intelligent search strategy for solving the flow-shop scheduling problem. Proc. IASTED Int. Conf. Software Engineering. Scottsdale, Arizona, USA, pp. 99-103.

Fan, J.P.O. 2002. An intelligent heuristic search method for flow-shop problems. Doctoral dissertation, Univ. of Wollongong, Wollongong, Australia.

Gheoweth, S.V.; and Davis, H.W. 1991. High performance A* search using rapidly growing heuristics. Proc. Int. Joint Conf. Artif. Intell. Sydney, Australia, pp. 198-203.

Hart, P.E.; Nilsson, N.J.; and Raphael, B. 1968. A formal basis for the heuristic determination of minimum cost paths. IEEE Trans. Systems Science and Cybernetics, Vol. SSC-4(2):100-7.

Hong, T.P.; and Chuang, T.N. 1998a. Fuzzy scheduling on two-machine flow shop. J. Intell. Fuzzy Syst. 6: 471-81.

Hong, T.P.; and Chuang, T.N. 1998b. Fuzzy CDS scheduling for flow shops with more than two machines. J. Intell.Fuzzy Syst. 6: 471-81.

Hong, T.P.; Huang, C.M.; and Yu, K.M. 1998. LPT scheduling for fuzzy tasks. Fuzzy Sets and Systems 97: 277-86.

Hong, T.P.; and Chuang, T.N. 1999. Fuzzy Palmer scheduling for flow shops with more than two machines. J. Info. Sci. Engineer.15: 397-406.

Hong, T.P.; and Wang, T.T. 1999. A heuristic Palmer-based fuzzy flexible flow-shop scheduling algorithm. Proc. IEEE Int. Conf. Fuzzy Systems 3: 1493-1497.

Hong, T.P.; Wang, C.L.; and Wang, S.L. 2000. A heuristic Gupta-based flexible flow-shop scheduling algorithm. Proc. IEEE Int. Conf. Systems, Man and Cybernetics 1: 319-22.

Ignall, E.; and Schrage, L.E. 1965. Application of the branch and bound technique to some flow shops scheduling problems. Oper. Res.. 13: 400-12.

Johnson, S.M. 1954. Optimal two- and three-stage production schedules with setup times included. Naval Res. Logistics Quart. 1(1), 61-68.

Kamburowski, J. 1997. Thenature of simplicity of Johnson's algorithm Omega-Int. J. Managem. Sci. 25: 581-4.

Korf, R.E. 1985a. Depth-firstiterative-deepening: an optimal admissible tree search. Art. Intell. 27: 97-109.

Korf, R.E. 1985b. Iterative-deepening A*: an optimal admissible tree search Proc. 9th Int. Joint Conf. Art. Intell., Morgan Kaufmann, Los Angeles, CA, USA, pp. 1034-6.

Korf, R.E. 1990. Real-time heuristic search. Art. Intel. 42: 189-211.

Korf, R.E. 1993. Linear-space best-first search. Art. Intell. 62(1): 41-78.

Lai, T.C. 1996. A note on heuristics of flow-shop scheduling. Oper.Res. 44: 648-52.

Lomnicki, Z. 1965. A branch and bound algorithm for the exact solution of three machine scheduling problem. Oper. Res. Quart. 16(1): 89-100.

McMahon, C.B.; and Burton, P.G. 1967. Flow-shop scheduling with the branch and bound method. Oper. Res.15: 473-81.

Pan, C.H. 1997. A study of integer programming formulations for scheduling problems. Int. J. Syst. Sci. 28(1): 33-41.

Pan, C.H.; and Chen, J.S. 1997. Scheduling alternative operations in two-machine flow-shops. J. Oper. Res. Soc. 48: 533-40.

Wang, C.G.; Chu, C.B.; and Proth, J.M. 1996. Efficient heuristic and optimal approaches for N/2/F/SIGMA-C-I scheduling problems. Int. J. Prod. Econ. 44: 225-37.

Winley, G.K.; and Fan, J.P.O. 2006a. Admissible heuristic functions for flow-shop problems. Int. J. Comput. Appl. Issue 3 (in press).

Winley, G.K.; and Fan, J.P.O. 2006b. Flow-shop problems: an improvement to an intelligent heuristic search algorithm. Asia-Pacific J. Oper. Res. (under review).

Zamani, M.R.; and Shue, L.Y. 1995. Developing an optimal learning search method for networks. Sci. Iranica 2: 197-206.

Zamani, R.; and Shue, L.Y. 1998. Solving project scheduling problems with a heuristic learning algorithm. J. Oper. Res. Soc. 49: 709-16.

Zamani, M.R. 2001. A high performance exact method for the resource-constrained project scheduling problem. Comp. Oper. Res. 28: 1387-14.

Appendix

Derivation of Heuristic Functions

From Fig 1

$S(\phi_{st}) \geq \max [b_t + \sum a, a_s + \sum b]$ and $T(\phi_{st}) \geq \max [S(\phi_{st}) + c_t, a_s + b_s + \sum c]$,

which means that:

$T(\phi_{st}) \geq a_s + b_s + \sum c$ (A1)

or, $T(\phi_{st}) \geq S(\phi_{st}) + c_t \geq b_t + c_t + \sum a$ (A2)

or, $T(\phi_{st}) \geq a_s + c_t + \sum b$. (A3)

From (A1) two heuristic functions H_1, H_9 are proposed:

$H_1 = \min[a_1 + b_1, a_2 + b_2, \ldots, a_n + b_n] + \sum c$; $H_9 = \min[a_1, a_2, \ldots, a_n] + \min[b_1, b_2, \ldots, b_n] + \sum c$.

H_3, H_7, H_8 are derived from (A2):

$H_3 = \min[v_1, v_2, \ldots, v_n] + \sum a$, where $v_k = \min[b_1 + c_1, b_2 + c_2, \ldots, b_{k-1} + c_{k-1}, b_{k+1} + c_{k+1}, \ldots, b_n + c_n]$; $H_7 = \min[b_1 + c_1, b_2 + c_2, \ldots, b_n + c_n] + \sum a$; $H_8 = \min[b_1, b_2, \ldots, b_n] + \min[c_1, c_2, \ldots, c_n] + \sum a$.

H_2 and H_6 are derived from (A3):

$H_2 = \min[a_1 + u_1, a_2 + u_2, \ldots, a_n + u_n] + \sum b$, where, $u_k = \min[c_1, c_2, \ldots, c_{k-1}, c_{k+1}, \ldots, c_n]$; $H_6 = \min[a_1, a_2, \ldots, a_n] + \min[c_1, c_2, \ldots, c_n] + \sum b$.

Two further heuristic functions H_4 and H_5 are proposed:

$H_4 = a_k + \min[c_1, \ldots, c_{k-1}, c_{k+1}, \ldots, c_n] + \sum b$, which is obtained by recognizing in H_3 that if $a_k = \min[a_1, a_2, \ldots, a_n]$ then the corresponding c_k can not be one of the possible values of c_t. $H_5 = \min[a_1 + c_1, a_2 + c_2, \ldots, a_n + c_n] + \sum b$, which assumes that there is no idle time on machine M_2 and that the job placed first in the sequence requires the least total time on machines M_1 and M_3.

Admissibility

Results and selected proofs related to the admissibility of the heuristic functions H_1, H_2, …, H_9, are presented:

R1. $H_1 \geq H_9$ and both are admissible.

R2. $H_2 \geq H_6$ and both are admissible.

R3. $H_3 \geq H_7 \geq H_8$ and each is admissible.

R4. If $a_k = \min[a_1, a_2, \ldots, a_n]$ and $c_k = c_m = \min[c_1, c_2, \ldots, c_n]$ then $H_4 \geq H_2 > H_6 = H_5$ and from R2 H_5 is admissible. Otherwise, $H_5 > H_2 = H_6 = H_4$ and from R2 H_4 is admissible.

Only proofs for R2, and the first part of R4, are given since the remaining proofs may be constructed in the same manner.

From (A3), $T(\phi_{st}) \geq a_s + c_t + \sum b$ for s, t = 1, 2, …, n with $s \neq t$ and so in particular, $T(\phi_{1t}) \geq a_1 + c_t + \sum b$, $T(\phi_{2t}) \geq a_2 + c_t + \sum b$, …, $T(\phi_{nt}) \geq a_n + c_t + \sum b$.

Hence, $T^*(\phi_{1t}) \geq \min[a_1 + c_2, a_1 + c_3, \ldots, a_1 + c_n] + \sum b$, $T^*(\phi_{2t}) \geq \min[a_2 + c_1, a_2 + c_3, \ldots, a_2 + c_n] + \sum b$, ..., $T^*(\phi_{nt}) \geq \min[a_n + c_1, a_n + c_2, \ldots, a_n + c_{n-1}, \ldots, a_n + c_n] + \sum b$ and $T^* = \min[T^*(\phi_{1t}), T^*(\phi_{2t}), \ldots, T^*(\phi_{nt})] \geq \min[a_1 + u_1, a_2 + u_2, \ldots, a_n + u_n] + \sum b = H_2 \geq \min[a_1, a_2, \ldots, a_n] + \min[c_1, c_2, \ldots, c_n] + \sum b = H_6$.

Consequently, $H_2 \geq H_6$ and both are admissible, which completes the proof of R2.

If $a_k = \min[a_1, a_2, \ldots, a_n]$ and $c_k = c_m = \min[c_1, c_2, \ldots, c_n]$ then $u_k > c_m$ and

$H_6 = a_k + c_m + \sum b$, $H_5 = \min[a_1 + c_1, \ldots, a_k + c_k, \ldots, a_n + c_n] + \sum b = a_k + c_m + \sum b$,

$H_2 = \min[a_1 + u_1, a_2 + u_2, \ldots, a_k + u_k, \ldots, a_n + u_n] + \sum b > a_k + c_m + \sum b = H_3 = H_5$,

$H_4 = a_k + u_k + \sum b$ and $H_2 = \min[a_1 + u_1 + \sum b, \ldots, H_4, \ldots, a_n + u_n + \sum b] \leq H_4$. Hence, $H_4 \geq H_2 > H_6 = H_5$ and, from R2, H_5 is admissible, which completes the proof of the first part of R4.

From results R1, R2, R3, it is seen that the heuristic functions $H_1, H_2, H_3, H_6, H_7, H_8, H_9$ are all admissible but in order to select the heuristic function among these that is the closest in value to the minimum makespan the choice should be made from among H_1, H_2, H_3.

H_4, H_5 are not considered as candidates for the best heuristic function to use because from result R4 it is seen that in the cases where they are admissible H_2 is as close or closer to the minimum makespan and in the cases where they are closer than H_2 no proof of their admissibility has been found.

Dominance

Referring to Table1, machine 1 dominates the other 2 machines if $\min[a_1, a_2, \ldots, a_n] \geq \max[b_1, b_2, \ldots, b_n]$ and $\min[a_1, a_2, \ldots, a_n] \geq \max[c_1, c_2, \ldots, c_n]$. Similar definitions apply if machine 2 or machine 3 is dominant.

In the case of a dominant machine results R5, R6, and R7, identify immediately which heuristic function among H_1, H_2, H_3 has the largest value and is the best to use in conjunction with IHSA*. Also, from R8 it is seen that the best heuristic function has a value which is significantly greater than the value of either of the other functions by $O(n^2)$, where n is the number of jobs.

R5. If machine 1 dominates then H_3 is the heuristic function with the largest value,

R6. If machine 2 dominates then H_2 is the heuristic function with the largest value,

R7. If machine 3 dominates then H_1 is the heuristic function with the largest value.

R8. If a machine is dominant then the best heuristic function has a value which is greater than the value of either of the other 2 heuristic functions by at least $(n - 1)(n - 2)$, where n is the number of jobs and $n \geq 3$.

The proofs for R5 and R8 are given noting that proofs for the other results may be constructed in the same manner.

Suppose machine1 dominates and $a_i \in [r_1, r_1 + w - 1]$, $b_i \in [s_1, s_1 + l - 1]$, $c_i \in [t_1, t_1 + d - 1]$ for i = 1, 2, …, n are random distinct positive integers from intervals of widths w, l, d respectively each greater than or equal to n (the number of jobs).

It follows that the minimum values of $\sum a$, $\sum b$, $\sum c$ are $nr_1 + 0.5n(n - 1)$, $ns_1 + 0.5n(n - 1)$, and $nt_1 + 0.5n(n - 1)$, respectively, and when these minimum values are attained $\min(a_i) = r_1$, $\min(b_i) = s_1$, $\min(c_i) = t_1$, $\max(a_i) = r_1 + n - 1$, $\max(b_i) = s_1 + n - 1$ and $\max(c_i) = t_1 + n - 1$.

Also, the maximum values of $\sum a$, $\sum b$, $\sum c$ are $n(r_1 + w) - 0.5n(n + 1)$, $n(s_1 + l) - 0.5n(n + 1)$, and $n(t_1 + d) - 0.5n(n + 1)$, respectively, and when these maximum values are attained $\min(a_i) = r_1 + w - n$, $\min(b_i) = s_1 + l - n$, $\min(c_i) = t_1 + d - n$, $\max(a_i) = r_1 + w - 1$, $\max(b_i) = s_1 + l - 1$, $\max(c_i) = t_1 + d - 1$.

Now,

$$H_3 = \min[v_1, v_2, \ldots, v_n] + \sum a \geq \min(b_i) + \min(c_i) + \min(\sum a) \quad \text{(A4)}$$

and similarly,

$$H_2 \leq \max(a_i) + \max(c_i) + \max(\sum b), \quad \text{(A5)}$$

$$H_1 \leq \max(a_i) + \max(b_i) + \max(\sum c). \quad \text{(A6)}$$

If (A4), (A5), (A6) are all true then,

$$H_3 \geq s_1 + l - n + t_1 + d - n + nr_1 + 0.5n(n - 1), \quad \text{(A7)}$$

$$H_2 \leq r_1 + n - 1 + t_1 + d - 1 + n(s_1 + l) - 0.5n(n + 1), \quad \text{(A8)}$$

$$H_1 \leq r_1 + n - 1 + s_1 + l - 1 + n(t_1 + d) - 0.5n(n + 1). \quad \text{(A9)}$$

From (A7) and (A8) we have:

$s_1 + l - n + t_1 + d - n + nr_1 + 0.5n(n - 1) - r_1 - n + 1 - t_1 - d + 1 - n(s_1 + 1) + 0.5n(n + 1) = s_1 - ns_1 + l - nl + nr_1 - r_1 + n^2 - 3n + 2 = (n - 1)[r_1 - (s_1 + l) + n - 2] \geq (n - 1)(n - 2) \geq 0$, for $n \geq 2$, and so $H_6 > H_2$ by a value at least as great as $(n - 1)(n - 2)$, for $n \geq 3$.

In a similar manner it follows from (A7) and (A9) that $H_3 > H_1$ by a value at least as great as $(n - 1)(n - 2)$ for $n \geq 3$, which completes the proof of R5 and R8.

It follows from R1, R2, R3 and R5, R6, R7 that if any one of the 3 machines is dominated by one or both of the other machines then there is no need to consider the heuristic function associated with the dominated machine when deciding which heuristic function to use in conjunction with IHSA*.