A Heuristic Search Algorithm for Flow-Shop Scheduling

Joshua Poh-Onn Fan
Graduate School of Business
University of Wollongong, NSW, Australia
E-mail: joshua@uow.edu.au

Graham K. Winley
Faculty of Science and Technology
Assumption University, Bangkok, Thailand
E-mail: gkwinley@scitech.au.edu

Keywords: Admissible heuristic function, dominance, flow-shop scheduling, optimal heuristic search algorithm

Received: October 25, 2007

This article describes the development of a new intelligent heuristic search algorithm (IHSA) which guarantees an optimal solution for flow-shop problems with an arbitrary number of jobs and machines provided the job sequence is constrained to be the same on each machine. The development is described in terms of 3 modifications made to the initial version of IHSA*. The first modification concerns the choice of an admissible heuristic function. The second concerns the calculation of heuristic estimates as the search for an optimal solution progresses, and the third determines multiple optimal solutions when they exist. The first 2 modifications improve performance characteristics of the algorithm and experimental evidence of these improvements is presented as well as instructive examples which illustrate the use of initial and final versions of IHSA*.*

Povzetek: Opisan je nov hevristični iskalni algoritem IHSA.*

1 Introduction

The optimal solution to the 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 on machines 1, 2, 3, …, m, in that order. The number of possible schedules is $(n!)^m$ and the general problem is NP-hard. For this general problem 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 [4]. Consequently, for the general problem with 2 or 3 machines there is an optimal solution where the jobs are processed in the same sequence on each machine and the optimal sequence is among only n! job sequences. However, in the optimal solution for the general problem with more than 3 machines the jobs are not necessarily processed in the same sequence on each machine. This article is concerned with the development of an algorithm (IHSA*) which is guaranteed to find an optimal solution for flow-shop scheduling problems involving an arbitrary number of jobs and machines where the problem is constrained so that the same job sequence is used on each machine.

Early research on flow-shop problems is based mainly on Johnson's theorem, which gives a procedure for finding an optimal solution with 2 machines, or 3 machines with certain characteristics [20], [21]. 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 [35]. Branch-and-bound methods use 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 [19], [29], [30], [36]. Heuristic methods have received significant attention [9], [18], [26], [27], [37], [40], [41], [42]. However, even the most powerful heuristic method to-date, the NEH heuristic developed by Nawaz et al. [31] fails to reach solutions within a reasonable bound of the optimal solution in some difficult problem cases [47]. A review of approaches by Zobolas et al. [47] indicates that there has been strong interest in artificial intelligence optimization methods referred to as metaheuristics including: Simulated Annealing [32], [34]; Tabu Search [11]; Genetic Algorithms, which may give an optimal solution but due to the evolutionary nature of this approach the computation time is unpredictable [2], [3], [5], [38]; Fuzzy Logic [13], [14], [15], [16], [17]; Ant Colony and Particle Swarm Optimization [28], [43]; Iterated Local Search [39]; and Differential Evolution [33]. The strong interest in metaheuristics generated the development of

hybrid approaches which combine different components of more than one metaheuristic [1].

An initial version of a new intelligent heuristic search algorithm (IHSA*) for flow-shop problems with an arbitrary number of jobs and machines and subject to the constraint that the same job sequence is used on each machine has been proposed in [6], [7], [8]. It is based on the Search and Learning A* algorithm presented in [44], [45], [46] which is a modified version of the Learning Real Time A* algorithm in [24], [25] which is, in turn, a modified version of the original A* algorithm [10], [12].

At the start of the search using IHSA* different methods are considered for computing estimates for the total time needed to complete all of the jobs on all of the machines assuming in turn that each of the jobs is placed first in the job sequence. It is shown that if there are m machines then there are m different methods that should be considered. Among the estimates associated with each method the smallest estimate is referred to as the value of the heuristic function that is associated with that method and it identifies the job that would be placed first in the job sequence at the start of the search if that method is used. If the value of the heuristic function does not exceed the minimum makespan for the problem then the heuristic function is said to be admissible and in such cases IHSA* is guaranteed to find an optimal solution provided the job sequence is the same on each machine. The proof of this result for IHSA* is given in [8] and is similar to that given in [22] and [23] in relation to the A* algorithm from which IHSA* is derived. The term "heuristic" is used in the title of IHSA* because the optimality of the algorithm and its performance depends on the selection of an appropriate admissible heuristic function at the start of the search and this function continues to guide the search to an optimal solution.

The purpose of this article is to describe the development of IHSA* which has occurred since the initial version was first presented in [6]. For simplicity of presentation the development is described in terms of problems involving an arbitrary number of jobs with 3 machines. However, the notations, definitions, proofs, and concepts presented may be extended to problems involving more than 3 machines if the job sequence is the same on each machine and these extensions are noted at the appropriate places throughout the presentation. Three significant modifications have been made to the initial version of IHSA*. The first concerns the choice of an admissible heuristic function at the start of the search. The second concerns the calculation of heuristic estimates as the search progresses, and the third determines multiple optimal solutions when they exist.

Following an introduction to the initial version of IHSA* each of the 3 modifications is presented. Experimental evidence of improvements in performance characteristics of the algorithm which result from the first 2 modifications is provided in the Appendix and discussed in section 5. Instructive examples are given to illustrate the initial and final versions of IHSA* and these have been limited to 3 machines in order to allow interested readers to familiarize themselves with the algorithm by reworking the examples by hand. The proofs of results related to the modifications are presented in the Appendix.

2 The initial version of IHSA*

Before presenting the initial version of the algorithm notations and definitions are introduced and the state transition process associated with IHSA* is described together with the features of search path diagrams which are used to illustrate the development of an optimal job sequence.

2.1 Notations and definitions

The following notations and definitions are introduced for a flow-shop problem involving n jobs $J_1, J_2, \ldots, J_n$ and 3 machines M_1, M_2, M_3.

O_{ij} is the operation performed on job J_i by machine M_j and there are 3n operations. For job J_i the processing times a_i, b_i, and c_i denote the times required to perform the operations O_{i1}, O_{i2}, and O_{i3}, respectively and these processing times are assumed to be non negative integers. If O_{ij} has commenced but has not been completed then p_{ij} represents the additional time required to complete O_{ij} and at the time when O_{ij} starts p_{ij} is one of the values among a_i, b_i, or c_i. The sequence $\phi_{st} = \{J_s, \ldots, J_t\}$ represents 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 of the jobs in ϕ_{st} are completed on machine M_2.

Using these notations and definitions Figure 1 illustrates the manner in which the operations associated with the job sequence ϕ_{st} are performed on the 3 machines.

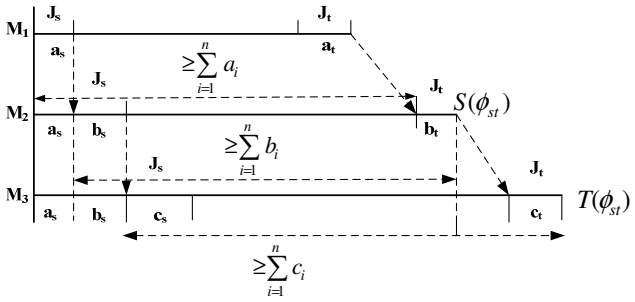

Figure 1: Processing the Job Sequence ϕ_{st}

A method for calculating an estimate of the total time to complete all of the n jobs on all of the 3 machines when job J_i is the first job in the sequence is given by $a_i + b_i + \sum_{i=1}^{n} c_i$. Then the heuristic function associated with this method is H_3 where,

$$H_3 = \min [a_1 + b_1, a_2 + b_2, \ldots, a_s + b_s, \ldots, a_n + b_n] + \sum_{i=1}^{n} c_i . \quad (1)$$

If $\min [a_1 + b_1, a_2 + b_2, \ldots, a_s + b_s, \ldots, a_n + b_n] = a_s + b_s$ then $H_3 = a_s + b_s + \sum_{i=1}^{n} c_i$ and job J_s would be

scheduled first in the job sequence. It is seen from Figure 1 and proved in the Appendix that H_3 is an admissible heuristic function and H_3 is the heuristic function that was used in the initial version of the IHSA*.

2.2 The state transition process

The procedure for developing the optimal job sequence using IHSA* proceeds by selecting an operation which may be performed next on an available machine. At the time that selection is made each of the 3n operations is in only one of 3 states: the *not scheduled* state; the *in-progress* state; or the *finished* state and the operation which is selected is among those in the *not scheduled* state. Operations not in the *finished* state are referred to as incomplete. A state transition occurs when one or more of the operations move from the *in-progress* state to the *finished* state and at any time the state level is the number of operations in the *finished* state.

IHSA* describes the procedure which takes the state transition process from one state level to the next and the development of the optimal job sequence is illustrated graphically using search path diagrams.

2.3 Search path diagrams

A search path diagram consists of nodes drawn at each state level with one of the nodes connected to nodes at the next state level. Each node contains 3 cells which are used to display information about operations on the machines M_1, M_2, and M_3, respectively. When a state transition occurs one of the nodes at the current state level is expanded and it is connected to the nodes at the next state level where each node represents one of the different ways of starting operations that are in the *not scheduled* state. At each of these nodes a cell is labeled with J_i to indicate that the operation O_{ij} is either in progress or is one of the operations that may start on M_j, and p_{ij} which is the time needed to complete J_i on M_j. A blank cell indicates that no operation can be performed on that machine at this time.

Near each node the heuristic estimate (h) associated with the node is recorded. The heuristic estimate is calculated using the heuristic function chosen in Step 1 of the algorithm in conjunction with the procedure used in Step 2. It is an estimate of the time required to complete the operation at the node identified by the procedure in Step 2 as well as all of the other 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 of the estimates for the nodes at that state level. Near this selected 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 is made between f and h' where h' is the minimum heuristic estimate at the preceding state level. Based on that comparison the search path either backtracks to the node expanded at the preceding state level or moves forward to the next state level. If backtracking occurs then the value of h' is changed to the current value of f and the search moves back to that node. If the path moves forward then the value of the edge cost (k) is recorded below 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. The final search path diagram represents the optimal solution and traces a path from one of the root nodes, where the minimum makespan is the value of h or f, to a terminal node where h = f = 0. The optimal job sequence can be read by recording the completed operations along the path from the root node to the terminal node.

2.4 IHSA* (initial version)

Step 1: At state level 0 expand the node identified by calculating the value of H_3 from (1), and move to the nodes at state level 1. If more than one node is identified then break ties randomly.

For example, if $H_3 = a_s + b_s + \sum_{i=1}^{n} c_i$ then the node with J_s and a_s recorded in the first cell is the node to be expanded.

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 at the current state level using Procedure 1 and proceed to Step 3.

Procedure 1 is described below.

Step 3: At the current state level select the node with the smallest heuristic estimate. If it is necessary then break ties randomly.

The smallest heuristic estimate is admissible and underestimates the minimum time required to complete all of the incomplete jobs on all of the machines.

Step 4: Calculate f = h + k where h is the smallest 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.

Procedure 1 is used in Step 2 to calculate a heuristic estimate for each node at the current state level:

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

$$h = \begin{cases} a_i + b_i + C_1; \text{ for } O_{i1} \text{ in the } \textit{not scheduled} \text{ state,} \\ p_{i1} + b_i + c_i + C_1; \text{ for } O_{i1} \text{ in the } \textit{in-progress} \text{ state,} \end{cases} \quad (2)$$

where C_1 is the sum of the values of c_k for all values of k such that O_{k1} is in the *not scheduled* state.

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

$$h = \begin{cases} b_i + C_2; \text{ for } O_{i2} \text{ in the } \textit{not scheduled} \text{ state,} \\ p_{i2} + c_i + C_2; \text{ for } O_{i2} \text{ in the } \textit{in-progress} \text{ state,} \end{cases} \quad (3)$$

where C_2 is the sum of the values of c_k for all values of k such that O_{k2} is in the *not scheduled* state.

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

$$h = \begin{cases} C_3; \text{ for } O_{i3} \text{ in the } \textit{not scheduled} \text{ state,} \\ p_{i3} + C_3; \text{ for } O_{i3} \text{ in the } \textit{in-progress} \text{ state,} \end{cases} \quad (4)$$

where C_3 is the sum of the values of c_k for all values of k such that O_{k3} is in the *not scheduled* state.

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

The following example illustrates the use of the initial version of IHSA* to solve the flow-shop problem given in Table 1. For instructional purposes the problem is deliberately simple with only 3 machines and 3 jobs because it is intended to provide readers with an opportunity to become familiar with the algorithm using an example that can be reworked easily by hand.

Table 1: Example of a 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

For illustrative purposes only the first search path diagram is presented in Figure 2.

At the start of the search $H_3 = 26$. In total 5 search path diagrams are required to find the optimal sequence $J_1J_2J_3$ with a minimum makespan of 26. Twenty nodes are expanded, 16 backtracking steps are required, and 43 steps of the algorithm are executed.

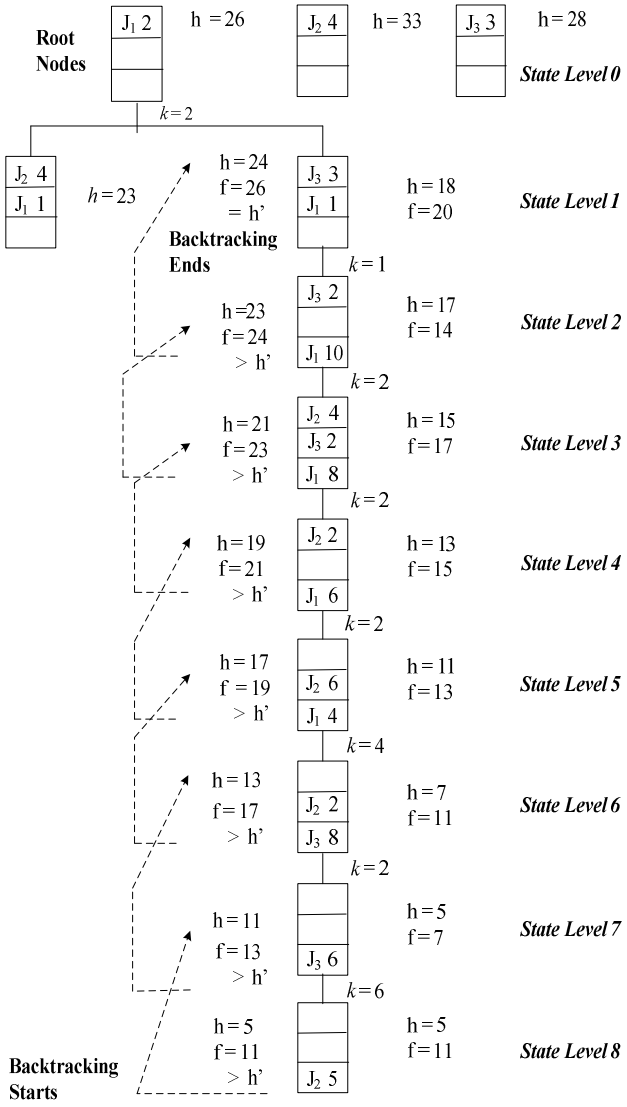

Figure 2: The First Search Path Diagram Using the Initial Version of IHSA*

3 Modifications to the initial version of IHSA*

The first modification determines the best heuristic function to use for a given problem and affects Step 1 of the algorithm. The second modification affects Procedure 1 used in Step 2 and the third modification affects Step 7 and enables multiple optimal solutions to be found when they exist.

3.1 A modification to step 1

In the Appendix a set of 6 heuristic functions are derived for the case of 3 machines and proofs of the admissibility of these functions are presented. It is shown that among this set of 6 heuristic functions the one which is admissible and has a value which is closest to the minimum makespan will be the one among H_1, H_2, and H_3 which has the largest value where,

$$\left.\begin{aligned} H_1 &= \min[b_1 + c_1, b_2 + c_2, \ldots, b_n + c_n] + \sum_{i=1}^{n} a_i, \\ H_2 &= \min[a_1 + u_1, a_2 + u_2, \ldots, a_n + u_n] + \sum_{i=1}^{n} b_i, \\ H_3 &= \min[a_1 + b_1, a_2 + b_2, \ldots, a_n + b_n] + \sum_{i=1}^{n} c_i, \end{aligned}\right\} \quad (5)$$

where: $u_1 = \min[c_2, c_3, \ldots, c_n]$; $u_k = \min[c_1, c_2, \ldots, c_{k-1}, c_{k+1}, \ldots, c_n]$, for $2 \le k \le n - 1$; and $u_n = \min[c_1, c_2, c_3, \ldots, c_{n-1}]$.

Choosing the admissible heuristic function among H_1, H_2, and H_3 with the largest value in the first step of the algorithm ensures that the search begins with an estimate of the minimum makespan that is not greater than it but is the closest to it. This choice is expected to reduce the need for backtracking at a subsequent stage of

the search since backtracking takes the search back to a previous node and increases the heuristic estimate at that node to a value which is admissible but closer to the minimum time required to complete all of the incomplete jobs on all of the machines.

Consequently, Step 1 in the initial version of IHSA* is modified and becomes:

Step1: At state level 0, from (5), choose the admissible heuristic function among H_1, H_2, and H_3 which has the largest value and if necessary break ties randomly. In the case where machine M_j dominates the other machines select H_j. Expand the node identified by the chosen admissible heuristic function and move to the nodes at state level 1. If more than one node is identified then break ties randomly.

The example in section 4 below illustrates the use of the modification to Step 1 in a simple problem which enables the reader to rework the example by hand. In the Appendix it is shown that in the particular case where machine M_j dominates the other machines then the best admissible heuristic function among H_1, H_2, and H_3 is H_j and it has a value which is greater than the value of either of the other two functions by at least $(n-1)(n-2)$ where n is the number of jobs. Thus in the case of a dominant machine in the first step of the algorithm there is no need to calculate each of the values of H_1, H_2, and H_3 in order to choose the one with the largest value. Instead, it is known that it will be H_j if machine M_j is the dominant machine.

In the case where there are m machines and $m > 3$ it is shown in the Appendix that the best admissible heuristic function to use in the first step of the algorithm is the one among $F_1, F_2, \ldots, F_m$ which has the largest value and if $m = 3$ then $F_1 = H_1$, $F_2 = H_2$, and $F_3 = H_3$. Experimental evidence of improvements in performance characteristics of the algorithm which result from using the modification to Step 1 is presented in the Appendix Table A1 and is discussed below in section 5.

3.2 A modification to step 2

The second modification to the initial version of IHSA* concerns the calculation of heuristic estimates at nodes on the search path when the search has commenced. It is based on the principle that when heuristic estimates for the nodes at the same state level are being calculated in Step 2 it is desirable to obtain the largest possible estimate at each of these nodes before selecting the node with the smallest estimate as the node to be expanded. The larger the value of this smallest estimate then the less likely it is that the search will need to backtrack and this is expected to improve the performance characteristics of the algorithm. As noted above, the use of Procedure 1 in Step 2 in the initial version of IHSA* gives a heuristic estimate for a node based on an operation in only one of the cells while operations in the other 2 cells are not taken into account.

The second modification affects Procedure 1 and involves calculating heuristic estimates h_1, h_2, and h_3 at a node for cells 1, 2, and 3, respectively. Then $\max[h_1, h_2, h_3]$ is used as the heuristic estimate (h) for the node. This is done for each node at the current state level and then, as before, in Step 3 the minimum estimate among these estimates identifies the node to be expanded. Consequently, Procedure 1 is replaced by the following Procedure 2:

In Step 2 of the algorithm for a node at the current state level,

$$\left.\begin{array}{l}\text{(a) For cell 1: If the cell is blank then } h_1 = 0.\\ \quad\text{Otherwise, } h_1 \text{ is given by (2).}\\ \text{(b) For cell 2: If the cell is blank then } h_2 = 0.\\ \quad\text{Otherwise, } h_2 \text{ is given by (3).}\\ \text{(c) For cell 3: If the cell is blank then } h_3 = 0.\\ \quad\text{Otherwise, } h_3 \text{ is given by (4).}\end{array}\right\} \quad (6)$$

Procedure 2 refers to calculations specified in Procedure 1 which incorporate currently the heuristic function H_3. However, (2), (3), and (4) are easily changed to incorporate H_1 or H_2 for problems where one of these functions has been selected using the modification to Step 1 of the algorithm. For example, if H_2 is used then (3) and (4) are not changed but (2) becomes,

$$h = \begin{cases} a_i + w_i + B_1; \text{ for } O_{i1} \text{ in the } \textit{not scheduled} \text{ state,} \\ p_{i1} + w_i + b_i + B_1; \text{ for } O_{i1} \text{ in the } \textit{in-progress} \text{ state,} \end{cases}$$

where, B_1 is the sum of the values of b_k for all values of $k \neq i$ such that O_{k1} is in the *not scheduled* state and w_i is the smallest value of c_k for all values of $k \neq i$ such that O_{k1} is in the *not scheduled* state.

Procedure 2 produces the same heuristic estimate as 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 cells 1 and 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. Consequently, using Procedure 2 will never produce an estimate that is less than the estimate produced by Procedure 1 and in practice the estimate using Procedure 2 is usually larger and leads to a reduction in backtracking.

Using Procedure 2 in the initial version of IHSA* modifies Step 2 and it 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.

If there are m machines and $m > 3$ then there are m cells at each node and an estimate is calculated for each cell by extending (6) and (2), (3), (4) to accommodate the heuristic function F_j (see Appendix) used in Step 1 of the algorithm.

The example in section 4 below illustrates the use of the modification to Step 2 in a simple problem which enables the reader to rework the example by hand. Experimental evidence of additional improvements in performance characteristics of the algorithm from using the modifications to Steps 1 and 2 together is presented in the Appendix Table A1 and is discussed below in section 5.

3.3 A modification to step 7

For some problems there are multiple optimal solutions and it is often important in practical situations to be able to find all of the optimal solutions since it may be required to find an optimal solution that also satisfies other criteria. For example, an optimal solution may be sought which also has the least waiting time for jobs that are queuing to be processed.

When IHSA* is implemented there are 2 situations which indicate the possible existence of multiple optimal solutions. The first situation occurs when there is more than 1 node at a state level with the smallest heuristic estimate. In this case the ties are broken randomly and one of the nodes is selected for expansion and the search continues and produces an optimal solution. At the completion of the search returning to that state level and selecting for expansion one of the other nodes which were not selected when the ties were broken may lead to a different optimal solution. The second situation occurs when at the completion of the search for an optimal solution one or more of the nodes at state level 0 has a heuristic estimate that is less than or equal to the minimum makespan. In this case returning to those nodes and beginning the search again may produce different optimal solutions.

The modification to the initial version of IHSA* that enables multiple optimal solutions to be determined affects Step 7. This modification is different from the previous 2 modifications in that it does not improve the performance characteristics of the algorithm but instead it is intended to find multiple optimal solutions if they exist.

Consequently, Step 7 becomes:

Step 7: If $f = 0$ and $h = 0$ then an optimal solution has been found. If along the path representing the optimal solution there is a node which was selected for expansion by breaking ties randomly among nodes at the same state level with the same minimum heuristic estimate then return to that state level and repeat from Step 2 ignoring any node that was selected previously for expansion as a result of breaking ties. If any of the values of h at root nodes (state level 0) is less than or equal to the minimum makespan then return to state level 0 and repeat from Step 2 ignoring root nodes that lead to a previous optimal solution. Otherwise, Stop.

4 The final version of IHSA*

The final version of IHSA* incorporates each of the 3 modifications:

Step1: At state level 0, from (5), choose the admissible heuristic function among H_1, H_2, and H_3 which has the largest value and if necessary break ties randomly. In the case where machine M_j dominates the other machines select H_j. Expand the node identified by the chosen admissible heuristic function and move to the nodes at state level 1. If more than one node is identified then break ties randomly.

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.

Step 3: At the current state level select the node with the smallest heuristic estimate. If it is necessary then break ties randomly.

Step 4: Calculate $f = h + k$ where h is the smallest 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 an optimal solution has been found. If along the path representing the optimal solution there is a node which was selected for expansion by breaking ties randomly among nodes at the same state level with the same minimum heuristic estimate then return to that state level and repeat from Step 2 ignoring any node that was selected previously for expansion as a result of breaking ties. If any of the values of h at root nodes (state level 0) is less than or equal to the minimum makespan then return to state level 0 and repeat from Step 2 ignoring root nodes that lead to a previous optimal solution. Otherwise, Stop.

If there are m machines and $m > 3$ then Steps 1 and Step 2 need to be modified in accordance with the discussion of this case presented in sections 3.1 and 3.2 above.

The simple instructive example which was used to illustrate the initial version of IHSA* (see Table 1) is used again to illustrate the final version of IHSA*. For this problem, from (5), $H_3 = 26 > H_1 = 19 > H_2 = 16$ and using the modification to Step 1 of the algorithm H_3 is used in Step 1 of the algorithm. Since no backtracking is necessary an optimal solution for the problem requires only 1 search path diagram which is shown in Figure 3 where the optimal solution has a minimum makespan of 26 and a job sequence J_1, J_2, J_3. At each node the search path diagram shows the estimates h_1, h_2, h_3 and the heuristic estimate for the node $h = max[h_1, h_2, h_3]$ which result from the use of the modification to Step 2 of the algorithm.

It is noted that the minimum heuristic estimate at state level 1 is 24 at both of the nodes at that state level. In Step 3 of the algorithm the tie was broken randomly and the node at which job J_2 is scheduled on machine M_1 was selected for expansion. In Step 7, although for simplicity a second search path diagram has not been drawn, the search returns to state level 1 and instead the

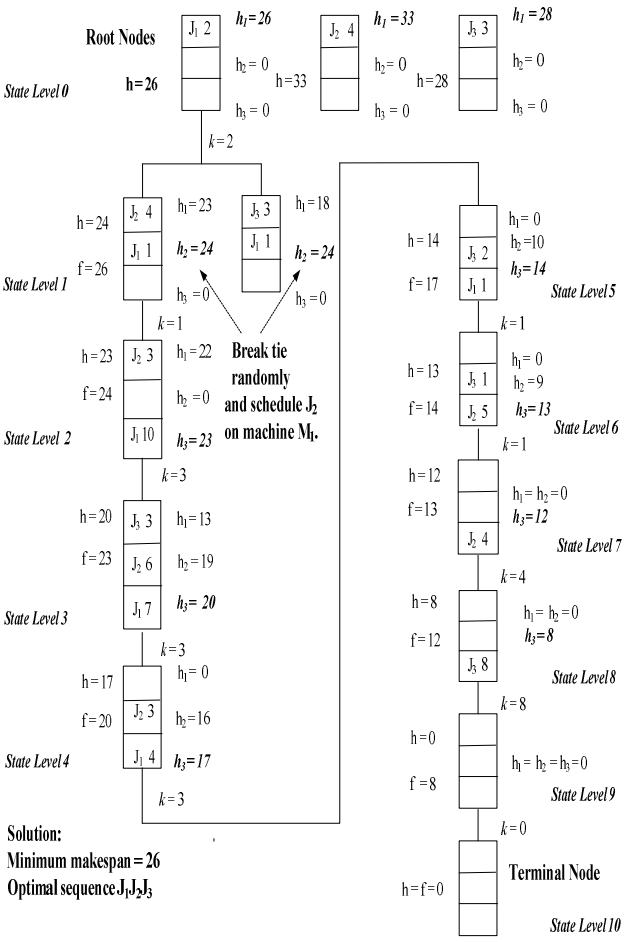

Figure 3: Search Path Diagram Using the Final Version of IHSA*

node at which job J_3 is scheduled on machine M_1 is expanded. This gives a second optimal solution where the job sequence is J_1, J_3, J_2.

5 Experimental evidence of improvements in performance

Experimental evidence of improvements in performance characteristics of IHSA* using the modifications to Steps 1 and 2 is presented in Appendix Table A1. The characteristics considered are: the number of nodes expanded; the number of backtracking steps required; and the number of steps of the algorithm executed.

In total 14 problems are considered involving: 3, 5 and 10 machines; and 3, 4, 10, 15, and 40 jobs. Each problem involving 3 machines was solved using the heuristic functions H_1, H_2, and H_3 in (5) which are the same as F_1, F_2, and F_3, respectively, when m = 3. Problems involving 5 and 10 machines were solved using their corresponding heuristic functions F_1, F_2, F_3, F_4, F_5 and F_1, F_2, F_3, …, F_{10}, respectively. The solutions enabled improvements in the performance characteristics resulting from the use of only the modification to Step 1 to be assessed. In addition, for each problem the solution was obtained using the modification to Step 1 together with the modification to Step 2. The performance characteristics associated with each of these solutions enabled an assessment of any further improvements in performance characteristics resulting from the inclusion of the modification to Step 2.

From Table A1 it is seen that for each problem regardless of the number of jobs and machines the modification to Step 1, which involves using the heuristic function with the largest value in Step 1, leads to improvements in all of the performance characteristics. Furthermore, in each problem using the modification to Step 1 together with the modification to Step 2, which affects the calculation of heuristic estimates as the search progresses, leads to further improvements in the performance characteristics.

6 Conclusion

Three modifications to the initial version of a new intelligent heuristic search algorithm (IHSA*) have been described. The algorithm guarantees an optimal solution for flow-shop problems involving an arbitrary number of jobs and machines provided the job sequence is the same on all of the machines.

The first modification affects 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 problem. For problems with an arbitrary number of jobs and 3 machines (M_1, M_2, M_3) a set of 6 possible functions is derived (H_1, H_2, …, H_6) and their admissibility is proved. It is shown that the function which has a value that is closest to the minimum makespan and is the best function to use in Step 1 of the algorithm is the function among H_1, H_2, and H_3 which has the largest value. In the particular case where one of the machines (M_j) dominates the other 2 machines the best function is H_j and there is no need to calculate the values of the other 2 functions. Furthermore, its value is greater than the value of either of the other 2 functions by at least $O(n^2)$ where n is the number of jobs. More generally, for problems with more than 3 machines (M_1, M_2, …, M_m) the best admissible heuristic function to use is the one among F_1, F_2, …, F_m with largest value and if machine M_j dominates the other machines then F_j is the best heuristic function. The proofs of these more general results may be obtained following the methods used in the proofs presented in the Appendix of the corresponding results for H_1, H_2, and H_3.

The second modification changes the procedure used in Step 2 of the initial version of the algorithm to determine heuristic estimates at nodes on the search path. The initial version determines a heuristic estimate at a node by considering an operation in only one of the cells at the node while operations in the other cells are not taken into account. The modified procedure determines a heuristic estimate at a node by selecting the largest of the separate estimates calculated for each cell at the node. The modified procedure never produces an estimate for a node that is smaller than the estimate produced by the procedure used in the initial version of the algorithm and in many cases it will be larger.

The first and second modifications ensure that at the start of the search and as the search progresses heuristic estimates are admissible and are as close as possible to the minimum time needed to complete all of the incomplete operations on all of the machines. This reduces the chance that the search will backtrack and improves the performance characteristics of the algorithm. Experimental evidence from problems involving various numbers of machines and jobs indicates that although the first modification produces improvements in performance characteristics of the algorithm these improvements are enhanced when the second modification is included.

The third modification relates to Step 7 of the algorithm and concerns problems where there are multiple optimal solutions. It enables all of the optimal solutions to be found and this is convenient for situations where additional criteria may need to be satisfied by an optimal solution.

This article has focussed on describing the development of the final version of IHSA*. However, there are several areas for future investigation including a comparison of the performance of the algorithm with other methods such as branch- and-bound methods and methods for pruning the search tree in order to improve memory management during implementation.

References

[1] Blum, C., Roli, A. "Metaheuristics in combinatorial optimization: overview and conceptual comparison," *ACM Comput. Surv.*, 35, 2003, 268-308.

[2] Chen, C.L., Neppalli, R.V., Aljaber, N. "Genetic algorithms applied to the continuous flow shop problem," *Computers and Industrial Engineering* 30: (4), 1996, 919-929.

[3] Cleveland, G.A., Smith, S.F. "Using genetic algorithms to schedule flow shop," Proceedings of 3rd Conference on Genetic Algorithms, Schaffer, D.(ed.), San Mateo: Morgan Kaufmann Publishing, 1989, 160-169.

[4] Conway, R.W., Maxwell, W.L., Miller, L.W. Theory of scheduling, Addison-Wesley, Reading Massachusetts, 1967.

[5] Eitler, O., Toklu, B., Atak, M., Wilson,J. "A genetic algorithm for flowshop scheduling problems," J. Oper. Res. Soc., 55, 2004, 830-835.

[6] Fan, J.P.-O. "The development of a heuristic search strategy for solving the flow-shop scheduling problem," Proceedings of the IASTED International Conference on Applied Informatics, Innsbruck, Austria, 1999, 516-518.

[7] Fan, J.P.-O. "An intelligent search strategy for solving the flow-shop scheduling problem," Proceedings of the IASTED International Conference on Software Engineering, Scottsdale, Arizona, USA, 1999, 99-103.

[8] Fan, J.P.-O. An intelligent heuristic search method for flow-shop problems, doctoral dissertation, University of Wollongong, Australia, 2002.

[9] Framinan, J.M, Ruiz-Usano, R., Leisten, R. "Sequencing CONWIP flow-shops: analysis and heuristic," Int. J. Prod. Res., 39, 2001, 2735-2749.

[10] Gheoweth, S.V., Davis, H.W. "High performance A* search using rapidly growing heuristics," Proceedings of the International Joint Conference on Artificial Intelligence, Sydney, Australia, 1991, 198-203.

[11] Grabowski, J., Wodecki, M. "A very fast tabu search algorithm for the permutation flowshop problem with makespan criterion," Comput. Oper. Res., 31, 2004, 1891-1909.

[12] Hart, P.E., Nilsson, N.J., Raphael, B. "A formal basis for the heuristic determination of minimum cost paths," IEEE Transactions on Systems Science and Cybernetics, Vol. SSC-4: (2), 1968, 100-107.

[13] Hong, T.P., Chuang, T.N. "Fuzzy scheduling on two-machine flow shop," Journal of Intelligent & Fuzzy Systems, 6: (4), 1998, 471-481.

[14] Hong, T.P., Chuang, T.N. "Fuzzy CDS scheduling for flow shops with more than two machines," Journal of Intelligent & Fuzzy Systems, 6: (4), 1998, 471-481.

[15] Hong, T.P., Chuang, T.N. "Fuzzy Palmer scheduling for flow shops with more than two machines," Journal of Information Science and Engineering, Vol.15, 1999, 397-406.

[16] Hong, T.P., Wang, T.T. "A heuristic Palmer-based fuzzy flexible flow-shop scheduling algorithm," Proceedings of the IEEE International Conference on Fuzzy Systems, Vol. 3, 1999, 1493-1497.

[17] Hong, T.P., Huang, C.M., Yu, K.M. "LPT scheduling for fuzzy tasks," Fuzzy Sets and Systems, Vol. 97, 1998, 277-286.

[18] Hong, T.P., Wang, C.L., Wang, S.L. "A heuristic Gupta-based flexible flow-shop scheduling algorithm," Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, Vol. 1, 2000, 319-322.

[19] Ignall, E., Schrage, L.E. "Application of the branch and bound technique to some flow shops scheduling problems," Operations Research, Vol. 13: (3), 1965, 400-412.

[20] Johnson, S.M. "Optimal two- and three-stage production schedules with setup times included," Naval Research Logistics Quarterly, 1: (1), 1954, 61-68.

[21] Kamburowski, J. "The nature of simplicity of Johnson's algorithm," Omega-International Journal of Management Science, 25: (5), 1997, 581-584.

[22] Korf, R.E. "Depth-first iterative-deepening: an optimal admissible tree search," Artificial Intelligence, Vol. 27, 1985, 97-109.

[23] Korf, R.E. "Iterative-deepening A*: an optimal admissible tree search," Proceeding of the 9th International Joint Conference on Artificial Intelligence, Los Angeles, California, 1985, 1034-1036.

[24] Korf, R.E. "Real-time heuristic search," Artificial Intelligence, Vol. 42, 1990, 189-211.

[25] Korf, R.E. "Linear-space best-first search," Artificial Intelligence, 62: (1), 1993, 41-78.
[26] Lai, T.C. "A note on heuristics of flow-shop scheduling," Operations Research, 44: (6), 1996, 648-652.
[27] Lee, G.C., Kim, Y.D., Choi, S. W. "Bottleneck-focused scheduling for a hybrid flow-shop," Int. J. Prod. Res., 42, 2004, 165-181.
[28] Liu, B., Wang, L., Jin, Y-H. "An effective PSO-based memetic algorithm for flow shop scheduling, "IEEE T. Syst. Man. CY. B.," 37, 2007, 18-27.
[29] Lomnicki, Z. "A branch and bound algorithm for the exact solution of three machine scheduling problem," Operational Research Quarterly, 16: (1), 1965, 89-100.
[30] McMahon, C.B., Burton, P.G. "Flow-shop scheduling with the branch and bound method," Operations Research, 15: (3), 1967, 473-481.
[31] Nawaz, M., Enscore Jr. E., Ham, I. "A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem," Omega-Int. J. Manage. S., 11, 1983, 91-95.
[32] Ogbu, F.A., Smith, D.K. "The application of the simulated annealing algorithm to the solution of the n/m/Cmax flowshop problem," Comput. Oper. Res., 17, 1990, 243-253.
[33] Onwubolu, G.C., Davendra, D. "Scheduling flow-shops using differential evolution algorithm," Eur. J. Oper. Res., 171, 2006, 674-692.
[34] Osman, I., Potts, C. "Simulated annealing for permutation flow shop scheduling," OMEGA, 17, 1989, 551-557.
[35] Pan, C.H. "A study of integer programming formulations for scheduling problems," International Journal of System Science, 28: (1), 1997, 33-41.
[36] Pan, C.H., Chen, J.S. "Scheduling alternative operations in two-machine flow-shops," Journal of the Operational Research Society, 48: (5), 1997, 533-540.
[37] Ravendran, C. "Heuristic for scheduling in flowshop with multiple objectives," Eur. J. Oper. Res., 82, 1995, 540-555.
[38] Ruiz, R., Maroto, C., Alcaraz, J. "Two new robust genetic algorithms for the flowshop scheduling problem," Omega-Int. J. Manage. S., 34, 2006, 461-476.
[39] Stutzle, T. "Applying iterated local search to the permutation flowshop problem," AIDA-98-04, TU Darmstadt, FG Intellektik, 1998.
[40] Taillard, E. "Some efficient heuristic methods for the flow shop sequencing problem," Eur. J. Oper. Res., 47, 1990, 65-74.
[41] Taillard, E. "Benchmarks for basic scheduling problems," Eur. J. Oper. Res., 64, 1993, 278-285.
[42] Wang, C.G., Chu, C.B., Proth, J.M. "Efficient heuristic and optimal approaches for N/2/F/SIGMA-C-I scheduling problems," International Journal of Production Economics, 44: (3), 1996, 225-237.
[43] Ying, K.C., Liao, C.J. "An ant colony system for permutation flow-shop sequencing," Comput. Oper. Res., 31, 2004, 791-801.
[44] Zamani, M.R., Shue, L.Y. "Developing an optimal learning search method for networks," Scientia Iranica, 2: (3), 1995, 197-206.
[45] Zamani, R., Shue, L.Y. "Solving project scheduling problems with a heuristic learning algorithm," Journal of the Operational Research Society, 49: (7), 1998, 709-716.
[46] Zamani, M.R. "A high performance exact method for the resource-constrained project scheduling problem," Computers and Operations Research, 28, 2001, 1387-14.
[47] Zobolas, G.I., Tarantilis, C.D., Ioannou, G. "Minimizing makespan in Permutation Flow Shop scheduling problems using a hybrid metaheuristic algorithm," Computers and Operations Research, 2008, doi:10.1016/j.cor.2008.01.007.

Appendix

Derivation of heuristic functions

The purpose is to develop heuristic functions suitable for use in IHSA*. In each case the objective is to develop a function which underestimates the minimum makespan (i.e. admissible). Six functions are developed and the proof of their admissibility is presented in the next section.

From Figure 1, $S(\phi_{st}) \geq \max [b_t +, a_s + \sum_{i=1}^{n} b_i]$ and $T(\phi_{st}) \geq \max [S(\phi_{st}) + c_t , a_s + b_s + \sum_{i=1}^{n} c_i]$ which means that:

$$T(\phi_{st}) \geq a_s + b_s + \sum_{i=1}^{n} c_i \text{ or,} \quad \text{(A1)}$$

$$T(\phi_{st}) \geq S(\phi_{st}) + c_t \geq b_t + c_t + \sum_{i=1}^{n} a_i \text{ or,} \quad \text{(A2)}$$

$$T(\phi_{st}) \geq a_s + c_t + \sum_{i=1}^{n} b_i . \quad \text{(A3)}$$

From (A1) two heuristic functions H_3 and H_6 are proposed:

$$H_3 = \min[a_1 + b_1, a_2 + b_2, \ldots, a_n + b_n] + \sum_{i=1}^{n} c_i \text{ and}$$

$$H_6 = \min[a_1, a_2, \ldots, a_n] + \min[b_1, b_2, \ldots, b_n] + \sum_{i=1}^{n} c_i .$$

The rationale for the development of H_3 is: select the job that will be finished on M_2 at the earliest possible time if it is placed first in the job sequence. When this job is finished on M_2 $\min[a_1 + b_1, a_2 + b_2, \ldots, a_n + b_n]$ units of time have elapsed and the additional time needed to complete all of the jobs on all of the machines will be at

least $\sum_{i=1}^{n} c_i$ units of time. Since $\min[a_1 + b_1, a_2 + b_2, \ldots, a_n + b_n] \geq \min[a_1, a_2, \ldots, a_n] + \min[b_1, b_2, \ldots, b_n]$ it follows that $H_3 \geq H_6$, which is therefore also a plausible heuristic function.

H_1 and H_5 are derived from (A2):

$H_1 = \min[b_1 + c_1, b_2 + c_2, \ldots, b_n + c_n] + \sum_{i=1}^{n} a_i$ and

$H_5 = \min[b_1, b_2, \ldots, b_n] + \min[c_1, c_2, \ldots, c_n] + \sum_{i=1}^{n} a_i$.

The rationale for the development of H_1 is: select the job which requires the least total amount of time on machines M_2 and M_3 (i.e. $\min[b_1 + c_1, b_2 + c_2, \ldots, b_n + c_n]$ units of time) and suppose that it is placed last in the job sequence which means that the earliest time that it can start on M_2 is after $\sum_{i=1}^{n} a_i$ units of time. Since $\min[b_1 + c_1, b_2 + c_2, \ldots, b_n + c_n] \geq \min[b_1, b_2, \ldots, b_n] + \min[c_1, c_2, \ldots, c_n]$ it follows that $H_1 \geq H_5$, which is therefore also a plausible heuristic function.

H_2 and H_4 are derived from (A3):

$H_2 = \min[a_1 + u_1, a_2 + u_2, \ldots, a_n + u_n] + \sum_{i=1}^{n} b_i$ and

$H_4 = \min[a_1, a_2, \ldots, a_n] + \min[c_1, c_2, \ldots, c_n] + \sum_{i=1}^{n} b_i$,

where: $u_1 = \min[c_2, c_3, \ldots, c_n]$; $u_k = \min[c_1, c_2, \ldots, c_{k-1}, c_{k+1}, \ldots, c_n]$ for $2 \leq k \leq n - 1$; and $u_n = \min[c_1, c_2, c_3, \ldots, c_{n-1}]$. The rationale for the development of H_2 is: consider each job in turn and suppose that it is placed first in the job sequence and then from among all of the other jobs select the one which requires the least amount of time on M_3. Now for each pair of jobs selected in this manner determine the pair that gives the least total time on M_1 and M_3. This total time plus the minimum total time required to finish all of the jobs on M_2 is the value of H_2. Also, $\min[a_1 + u_1, a_2 + u_2, \ldots, a_n + u_n] \geq \min[a_1, a_2, \ldots, a_n] + \min[u_1, u_2, \ldots, u_n] = \min[a_1, a_2, \ldots, a_n] + \min[c_1, c_2, \ldots, c_n]$ and it follows that $H_2 \geq H_4$, which is therefore also a plausible heuristic function.

Admissibility

Results and selected proofs related to the admissibility of the heuristic functions H_1, H_2, H_3, H_4, H_5, and H_6 are presented:

R1. $H_3 \geq H_6$ and both are admissible.

R2. $H_2 \geq H_4$ and both are admissible.

R3. $H_1 \geq H_5$ and both are admissible.

Only a proof for ***R2*** is given since the remaining proofs may be constructed in the same manner.

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

Hence, if $T^*(\phi_{st})$ denotes the earliest time at which any job sequence which starts with job J_s is completed on M_3 then $T^*(\phi_{1t}) \geq \min[a_1 + c_2, a_1 + c_3, \ldots, a_1 + c_n] + \sum_{i=1}^{n} b_i$, $T^*(\phi_{2t}) \geq \min[a_2 + c_1, a_2 + c_3, \ldots, a_2 + c_n] + \sum_{i=1}^{n} b_i$, …, $T^*(\phi_{nt}) \geq \min[a_n + c_1, a_n + c_2, \ldots, a_n + c_{n-1}, \ldots, a_n + c_n] + \sum_{i=1}^{n} b_i$ and the minimum makespan $T^* = \min[T^*(\phi_{1t}), T^*(\phi_{2t}), \ldots, T^*(\phi_{nt})] \geq \min[a_1 + u_1, a_2 + u_2, \ldots, a_n + u_n] + \sum_{i=1}^{n} b_i = H_2 \geq \min[a_1, a_2, \ldots, a_n] + \min[c_1, c_2, \ldots, c_n] + \sum_{i=1}^{n} b_i = H_4$. Consequently, $H_2 \geq H_4$ and both are admissible.

From the results *R1*, *R2*, and *R3* it is seen that the heuristic functions H_1, H_2, H_3, H_4, H_5, and H_6 are all admissible. However, in order to select the heuristic function among these that is the closest in value to the minimum makespan (i.e. the best to use in Step1 of IHSA*) the choice should be made from among only H_1, H_2, and H_3 because the function among these 3 which has the largest value is admissible and has a value which is larger than any of the other 5 admissible functions. Consequently, in Step1 of IHSA* the values of H_1, H_2, and H_3 are calculated and the function with the largest value is selected for use.

Dominance

Machine M_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 $\min[a_1, a_2, \ldots, a_n] \geq \max[c_1, c_2, \ldots, c_n]$ and similar definitions apply if machine M_2 or machine M_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, and H_3 has the largest value and is the best to use in IHSA*. Also, from *R8* it is seen that the best heuristic function has a value which is greater than the value of either of the other functions by $O(n^2)$ where n is the number of jobs.

R5. If machine M_1 dominates then H_1 is the heuristic function with the largest value,

R6. If machine M_2 dominates then H_2 is the heuristic function with the largest value,

R7. If machine M_3 dominates then H_3 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 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. Throughout these proofs $\min(a_i) = \min[a_1, a_2, \ldots, a_n]$, $\min(b_i) = \min[b_1, b_2, \ldots, b_n]$, $\min(c_i) = \min[c_1, c_2, \ldots, c_n]$, $\max(a_i) = \max[a_1, a_2, \ldots, a_n]$, $\max(b_i) = \max[b_1, b_2, \ldots, b_n]$, and $\max(c_i) = \max[c_1, c_2, \ldots, c_n]$.

Suppose machine M_1 dominates and for i = 1, 2, 3, …, n: $a_i \in [r_1, r_1 + w - 1]$; $b_i \in [s_1, s_1 + l - 1]$; and $c_i \in [t_1, t_1 + d - 1]$ are distinct non negative integers from intervals of widths w, l, and d, respectively, each greater than or equal to n (the number of jobs).

It follows that the minimum values of $\sum_{i=1}^{n} a_i$, $\sum_{i=1}^{n} b_i$, $\sum_{i=1}^{n} c_i$ 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 $\max(c_i) = t_1 + n - 1$ for i = 1, 2, 3, …, n.

Also, the maximum values of $\sum_{i=1}^{n} a_i$, $\sum_{i=1}^{n} b_i$, $\sum_{i=1}^{n} c_i$ 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, $H_1 = \min[b_1 + c_1, b_2 + c_2, \ldots, b_n + c_n] + \sum_{i=1}^{n} a_i$

$$\geq \min(b_i) + \min(c_i) + \min(\sum_{i=1}^{n} a_i) \qquad \text{(A4)}$$

and similarly,

$$H_2 \leq \max(a_i) + \max(c_i) + \max(\sum_{i=1}^{n} b_i) \qquad \text{(A5)}$$

and

$$H_3 \leq \max(a_i) + \max(b_i) + \max(\sum_{i=1}^{n} c_i). \qquad \text{(A6)}$$

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

$$H_1 \geq s_1 + l - n + t_1 + d - n + nr_1 + 0.5n(n-1), \qquad \text{(A7)}$$

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

$$H_3 \leq r_1 + n - 1 + s_1 + l - 1 + n(t_1 + d) - 0.5n(n+1). \qquad \text{(A9)}$$

From (A7) and (A8),

$s_1 + l - n + t_1 + d - n + nr_1 + 0.5n(n-1) - r_1 - n + 1 - t_1 - d + 1 - n(s_1 + 1) + 0.5n(n+1) = s_1 - ns_1 + l - n_l + nr_1 - r_1 + n^2 - 3n + 2 = (n-1)[r_1 - (s_1 + l) + n - 2] \geq (n-1)(n-2) \geq 0$, for $n \geq 2$, and so H_1 is greater than H_2 by a value which is at least $(n-1)(n-2)$, for $n \geq 3$.

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

The best admissible heuristic function for an arbitrary number of machines

For the case where there are more than 3 machines there is a need to change the notation used previously to represent the time that each operation $O_{i,j}$ requires on each machine so that $t_{i,j}$ is the number of units of time required by job J_i on machine M_j.

If there are m machines then the best admissible heuristic function will be the one with the largest value among the set of m functions $F_1, F_2, F_3, \ldots, F_m$ where,

$$F_j = \begin{cases} \min[\sum_{i=2}^{m} t_{1,i}, \sum_{i=2}^{m} t_{2,i}, \ldots, \sum_{i=2}^{m} t_{n,i}] + \sum_{i=1}^{n} t_{i,1}, \text{ for } j=1, \\ \min[\sum_{i=1}^{j-1} t_{1,i} + u_{j,1}, \sum_{i=1}^{j-1} t_{2,i} + u_{j,2}, \ldots, \sum_{i=1}^{j-1} t_{n,i} + u_{j,n}] \\ \quad + \sum_{i=1}^{n} t_{i,j}, \text{ for } 2 \leq j \leq m, \end{cases}$$

where, for $2 \leq j \leq m - 1$,

$$u_{j,k} = \begin{cases} \min[\sum_{i=j+1}^{m} t_{2,i}, \sum_{i=j+1}^{m} t_{3,i}, \ldots, \sum_{i=j+1}^{m} t_{n,i}], \text{ for } k = 1, \\ \min[\sum_{i=j+1}^{m} t_{1,i}, \sum_{i=j+1}^{m} t_{2,i}, \ldots, \sum_{i=j+1}^{m} t_{k-1,i}, \sum_{i=j+1}^{m} t_{k+1,i}, \ldots, \\ \quad \sum_{i=j+1}^{m} t_{n,i}], \text{ for } 2 \leq k \leq n-1, \\ \min[\sum_{i=j+1}^{m} t_{1,i}, \sum_{i=j+1}^{m} t_{2,i}, \ldots, \sum_{i=j+1}^{m} t_{n-1,i}], \text{ for } k = n, \end{cases}$$

and $u_{m,k} = 0$, for k = 1, 2, 3, …, n.

For a problem with m machines where m > 3 and the job sequence is the same on each machine the function among $F_1, F_2, F_3, \ldots, F_m$ with the largest value is selected in Step 1 of IHSA*.

If m = 3 then using $t_{1,i} = a_i$, $t_{2,i} = b_i$, and $t_{3,i} = c_i$ for i = 1, 2, 3, …, n and representing $u_{j,1}$, $u_{j,k}$, and $u_{j,n}$ simply by u_1, u_k, and u_n, respectively, the 3 admissible heuristic functions H_1, H_2, and H_3 in (5) which have been used throughout the description of the development of IHSA* are given by F_1, F_2, and F_3, respectively.

Experimental evidence of improvements in performance characteristics

Table A1: Performance of IHSA*: Modification to Step 1 compared to Modifications to Steps 1 and 2.

Note: For each problem: (a) the highlighted first row, associated with the use of the modification to Step 1, indicates the performance characteristics using the best heuristic function; (b) the highlighted last row, associated with the use of modifications to Steps 1 & 2, indicates the performance characteristics when the best heuristic function is used together with the modification to Step 2.

No. of Machines	Number of Jobs	Problem	Modification Used	Heuristic Function	Value of Heuristic Function	Performance Characteristics			Minimum Makespan
						Nodes Expanded	Backtracks	Algorithm Steps	
3	3	1	1	**H1**	**17**	**13**	**0**	**10**	17
				H3	10	22	11	33	
				H2	9	21	12	34	
			1& 2	**H1**	**17**	**10**	**0**	**7**	
		2	1	**H2**	**24**	**13**	**6**	**22**	24
				H3	15	36	39	88	
				H1	13	36	37	84	
			1& 2	**H2**	**24**	**8**	**2**	**17**	
		3	1	**H3**	**28**	**16**	**1**	**14**	28
				H2	20	17	3	16	
				H1	18	17	3	16	
			1& 2	**H3**	**28**	**13**	**0**	**10**	
	4	4	1	**H1**	**23**	**35**	**20**	**52**	25
				H3	19	50	30	72	
				H2	14	57	37	82	
			1& 2	**H1**	**23**	**35**	**15**	**42**	
		5	1	**H2**	**22**	**32**	**27**	**66**	23
				H3	19	79	70	152	
				H1	15	80	79	170	
			1& 2	**H2**	**22**	**31**	**21**	**54**	
		6	1	**H3**	**24**	**66**	**53**	**109**	27
				H2	21	69	63	138	
				H1	20	71	64	140	
			1& 2	**H3**	**24**	**63**	**43**	**96**	
	15	7	1	**H1**	**26**	**20**	**2**	**22**	28
				H3	18	24	10	30	
				H2	16	26	15	36	
			1& 2	**H1**	**26**	**18**	**0**	**10**	
		8	1	**H2**	**28**	**20**	**9**	**38**	31
				H3	20	38	40	90	
				H1	16	40	45	98	
			1& 2	**H2**	**28**	**16**	**4**	**25**	
		9	1	**H3**	**29**	**19**	**2**	**22**	30
				H2	20	22	4	28	
				H1	19	25	4	29	
			1& 2	**H3**	**29**	**16**	**0**	**18**	
	40	10	1	**H1**	**43**	**50**	**12**	**70**	45
				H3	38	53	30	105	
				H2	35	70	38	113	
			1& 2	**H1**	**43**	**44**	**6**	**55**	
		11	1	**H2**	**42**	**49**	**30**	**60**	43
				H3	40	50	32	85	
				H1	37	66	45	115	
			1& 2	**H2**	**42**	**32**	**18**	**52**	
		12	1	**H3**	**45**	**68**	**46**	**115**	50
				H2	38	77	58	152	
				H1	35	90	93	205	
			1& 2	**H3**	**45**	**54**	**30**	**84**	
5	15	13	1	**F1**	**37**	**42**	**12**	**65**	39
				F2	30	60	15	93	
				F3	27	72	18	104	
				F4	25	82	25	126	
				F5	23	97	32	150	
			1& 2	**F1**	**37**	**30**	**5**	**50**	
10	10	14	1	**F3**	**50**	**35**	**20**	**63**	52
				F2	48	37	23	83	
				F4	46	41	28	89	
				F1	45	43	30	91	
				F6	42	45	33	95	
				F5	40	52	34	107	
				F9	38	60	40	123	
				F7	32	69	45	135	
				F8	32	72	48	137	
				F10	30	81	51	151	
			1& 2	**F3**	**50**	**30**	**2**	**40**	

Table A1: Performance of IHSA*