

Hierarchical Composition Heuristic for Sequence Dependent Scheduling Problems

Ceyhun O. Ozgur e-mail: ceyhun.ozgur@valpo.edu Tel: (219) 464-5178

Lihui Bai e-mail: lihui.bai@valpo.edu Tel: (219) 464-5188

College of Business Administration

Valparaiso University

Valparaiso, Indiana 46383

Kenneth A. Dunning e-mail: dunning@uakron.edu Tel: (330) 972-7039

College of Business Administration

The University of Akron

Akron, Ohio 44325-4801

ABSTRACT

This paper briefly reviews the scheduling procedures with sequence dependent setups developed by previous research. We present a procedure for single machines with asymmetric sequence dependent setups where the objective is to minimize the total setup time. We also illustrate the application of the procedure for an existing industrial problem. The procedure is named Hierarchical Composition Heuristic. It is a two-stage procedure that takes advantage of the natural product groupings. Therefore, it should be a very valuable scheduling tool for production environments that manufacture groups of items. A computational study of the procedure is reported along with general guidelines for application in similar scheduling environments.

Key words: traveling salesman problem, sequence dependent setups, heuristic, combinatorial optimization

INTRODUCTION

The recent advances in technology and increases in global competition necessitate firms to be more efficient in managing their operations. The improvement in efficiency can be achieved by reducing setup times. Setup time remains an important scheduling consideration for industries such as metal processing, commercial printing, plastics and many others. In the production scheduling literature, it is frequently assumed that the changeover time between jobs is independent of the processing sequence. Although the assumption of sequence independence is frequently made, it does not hold in some industrial applications. If the setups are sequence independent, the sequence of jobs at a work center does not impact the total setup time for that work center. If the setup times are sequence dependent, the setup time of the current job depends on the previous job. The sequence of jobs can significantly impact the total setup time which in turn can have serious implications on the efficiency of operations. Thus, if the jobs have sequence dependent setups, an appropriate scheduling procedure that considers this special characteristic should be developed to improve the overall efficiency of the machine center. Many of the existing scheduling procedures with sequence dependent setups assume that setup time between two jobs is symmetric. Symmetry between job 1 and job 2 means that it takes the same amount of time to changeover from job 1 to job 2 as it does to changeover from job 2 to job 1. Even with this simplifying assumption, existing optimization algorithms do not provide an optimal solution to large problems in a reasonable amount of time. This paper develops a solution procedure for a single machine with sequence dependent setups and asymmetric changeover times, where changing from job 1 to job 2 is different than changing from job 2 to job 1.

One of the most common combinatorial problems faced in sequence dependent scheduling environments is the Traveling Salesman Problem (TSP), where the objective is to find a minimum distance tour that visits each of n cities exactly once and returns to the city of origin. The input data for the problem is an input matrix of distances between pairs of cities. In scheduling applications, time or cost is used as a measure instead of distance, and each city corresponds to a job. In the past, researchers have developed many algorithms and heuristics to solve the TSP and found practical applications for Traveling Salesman type scheduling problems in the industry. However, it is also a well known fact that the TSP is NP-complete, and no polynomially bounded algorithms that provide the optimal solution have been developed. Both branch and bound as well as dynamic programming have been used to construct non-polynomially bounded algorithms for the TSP. Because of the NP-completeness of the TSP, the algorithms that will find the

shortest sequence of jobs take an unreasonably long time to execute in a scheduling environment subject to frequent scheduling changes. Therefore, some researchers have resorted to heuristics that can find good, feasible solutions very quickly. The heuristic presented later in this paper is called Hierarchical Composition (HICOM). It provides a feasible sequence of items for a single machine with sequence dependent setups and asymmetric setup times. It is a simple and intuitive heuristic that can be understood by practitioners. It is a flexible procedure that can be applied to problems with symmetric or asymmetric setup times between pairs of jobs. In addition, it can also be used as a clustering method for the items produced on the machine. In a later section, we will show that it will quickly provide a good solution to the scheduling problem. It is especially appropriate to apply this heuristic to a machine in a dynamic environment that produces families of items because the procedure will take advantage of product group structures and can quickly respond to changes in the master schedule.

PROBLEM SETTING

The solution procedure is developed for an automatic cable assembly machine for a leading firm in the business of remote mechanical push-pull control systems. The cable assembly system is a non-synchronous conveyor that transports a coiled cable from station-to-station until all of the assembly operations are completed. There are two, similar stand-alone cable assembly machines, each one producing cables with different characteristics. One of the major differences between the two machines from a scheduling perspective is that the first machine has symmetric sequence dependent setup times, while the second machine experiences asymmetric sequence dependent setup times. There are 14 workstations, of which 12 require setups. At each workstation a polyethylene pallet containing the coiled cable is stopped, raised off the conveyor rolls, and rigidly fixtured for the operations to be performed at the workstation. The cable assembly system performs all operations unattended with the exception of supply and maintenance of necessary parts for the workstations. When the automated assembly machine starts to manufacture a different item (cable), the parts at some of the workstations must be changed, thereby incurring a setup for each of those workstations. The total setup time between any two items is the sum of the setup times over all of the workstations. Management felt that the items produced on the cable assembly system could be put into groups where the setup time between items in the same group would be significantly smaller than items in different groups.

THE HIERARCHICAL COMPOSITION (HICOM) HEURISTIC

The HICOM heuristic utilizes an item combination and hierarchical group building mechanism similar to tour building for the TSP. Groups are built on the basis of similarity of setups between pairs of items/groups. In addition to utilizing this procedure as a scheduling tool, one can also use it as a clustering method to determine group membership. The more distinct the groups are, the greater the effectiveness of the procedure. The objective of the sequencing procedure is to minimize the total setup time of all items produced on the assembly system for a given time period. The procedure consists of two phases and requires the knowledge or estimate of setup times between all pairs of items. Let a square $n \times n$ matrix (S) represent the setup times between all n items. S is an asymmetric matrix where $s_{ij} \neq s_{ji}$, and $s_{ii} = \infty \forall i$. Technically, even though the setup time for s_{ii} is zero, it is set to infinity in order to block the variable from coming into the solution. Note that distance s_{12} represents the setup time associated with changing from producing item 1 to producing item 2.

Once we have a setup time matrix, the HICOM heuristic can be executed. Initially, the setup times in the entire setup time matrix S are sorted in ascending order of their setup times. From the sorted list, we select the smallest setup time s_{ij} . Selection of this setup time involves scheduling item i and item j in the i - j order. In the sequence of the i - j order, we refer herein product i as the tail node and product j as the head node. Since we are committed to change from product i to product j , elimination of row i ensures that we cannot changeover from item i to any other item. Similarly, elimination of column j guarantees that we cannot switch to j from any other item. In addition, setup time s_{ij} can also be eliminated from further consideration since items i and j have been scheduled in the i - j order. After the row and column eliminations are performed, the remaining smallest setup time is selected from the sorted S matrix. The new row i and new column j and setup time s_{ij} are eliminated again. Every time a new s_{ij} value is selected from the sorted list of remaining items in matrix S , a new iteration begins. After making the proper eliminations, we continue to sequence product i and j . This sequencing procedure leads to the formation of product groups. It is

possible that both product i and product j have already been assigned in a previous iteration to two different groups. When this situation occurs, no additional assignments are made since product i and j have already been assigned, and the remaining feasible, smallest setup time is selected from the sorted S matrix. The row and column elimination procedure does not prevent product i and product j from having been selected in previous iterations and classified into the same group. Due to the row and column elimination procedure administered, this can only occur if product j is at the beginning and product i is at the end of the sequence of the existing group. When this situation occurs (product i and j have been selected in previous iterations and are classified into the same group), no new assignments are made, the remaining minimum setup time is selected from matrix S , and the procedure continues.

Every time a new s_{ij} is selected, it is also possible that one of the products might belong to an existing group. If product i belongs to an existing group from a previous iteration, then its placement in the sequence of the existing group must be at the end of that group. This is because product i could have been selected in a previous iteration only as a head node due to the elimination methodology of rows and columns. In this situation, we sequence product j after product i at the end of the sequence. Similarly, if product j belongs to an existing group from a previous iteration, then it must be placed at the beginning of the sequence in that group. Thus, we sequence product i before product j at the beginning of the sequence. This elimination procedure will continue until all the rows and the columns in the S matrix are eliminated. Elimination of all the rows and the columns concludes the first phase of the HICOM heuristic. Finally, we keep two sets of nodes: the set of head nodes and the set of tail nodes for all the groups. Whenever a new s_{ij} is selected, we eliminate s_{jq} and s_{pi} where product q is any tail node and p is any head node in their current sets. Doing so, we eliminate the possibility that sub-loops may form in later iterations. At each iteration, if a group is expanded or a new group is formed, we update the sets of head and tail nodes accordingly.

At the end of the first phase, items are divided into m groups with their respective sequences. The objective of the second phase is to combine the groups to minimize the total setup time. In developing the group changeover matrix " G ", the value of g_{ij} must be determined for all i, j where g_{ij} is the changeover time from group i to group j . For the same reasons mentioned earlier, $g_{ii} = \infty$ for all i . If i and j are not equal, changeover time from the product at the end of the sequence in group i to the product at the beginning of the sequence in group j constitutes the setup time between group i and group j . The determination of the group changeover time matrix " G ", a sequencing procedure similar to the one in Phase 1, combines the groups instead of the items. Elimination of all rows and columns in matrix G completes the first iteration of the second phase of this procedure. The second iteration of the second phase begins by updating G so that the new matrix reflects the changeover times between the new reduced set of groups obtained at the end of the first iteration of the second phase. This procedure will continue until all the items are combined into one group. When the number of groups is reduced to one, the procedure terminates with a feasible sequence of all items, i.e., the heuristic solution by HICOM.

This procedure can be classified as a hierarchical clustering method. Many of these hierarchical clustering techniques use a series of mergers. These techniques start with as many clusters as number of items. First, the two most similar items are combined. Then the next two similar items, or an item and a group, or two groups are combined, and so on. One of the common methods of joining clusters (groups) is called the single linkage rule (SLR) [7]. This method combines the two groups with the smallest distance between them. The distance between the two groups is considered to be the distance between their closest points. The procedure of merging items and groups is similar to HICOM. The major difference between HICOM and the single linkage algorithm is that SLR does not sequence the items.

During the second phase of HICOM, each iteration produces new and smaller sets of groups. One of these sets of groups can determine the group membership of all items for classification purposes. Another advantage of this method is that group membership is not pre-specified because the clusters are determined by the heuristic. Notation and steps of the Hierarchical Composition (HICOM) Heuristic follow:

Notation

Name	Description	Name	Description
$S_{n \times n}$	a square matrix of setup times between products that are sequence-dependent	s_{ij}	the setup time from product (row) i to product (column) j
n	the number of products	m	the number of groups
t	the iteration index for phase 2	$G_{m \times m}$	a square matrix of setup times between groups.
k	the iteration index for phase 1	g_{ij}	the setup (changeover) time from group (row) i to group (column) j

PHASE 1

Step 1 (initialization): Let $k=1$ and the changeover matrix $S_k = S$. Let $T = \emptyset$ and $H = \emptyset$.

Step 2 (sorting): Sort all the setup times in matrix S_k in ascending order. Select the smallest setup time $(s_{ij})^k$ from the setup time matrix S_k . Let $(i)^k$ and $(j)^k$ be the row and column indices, respectively.

Step 3 (elimination):

(a) Eliminate all the setup times in row $(i)^k$ and column $(j)^k$ and the setup time $(s_{ij})^k$ from S_k .

(b) Eliminate all the setup times for element $(S_{pq})^k \in \{S_{pq} \mid p \in H \text{ and } q = (i)^k, \text{ or } p = (j)^k \text{ and } q \in T\}$.

Step 4 (grouping):

(a) If product $(i)^k$ is already in an existing group, link product $(j)^k$ to the end of the sequence in that existing group if $(j)^k$ does not already exist earlier in the sequence; update $H = H \setminus \{(i)^k\} \cup \{(j)^k\}$.

(b) If product $(j)^k$ is already in an existing group, attach product $(i)^k$ at the beginning of the sequence in that existing group if $(i)^k$ does not already exist later in the sequence; update $T = T \setminus \{(j)^k\} \cup \{(i)^k\}$.

(c) If neither $(i)^k$ nor $(j)^k$ is in an existing group, place both products in a new group in the $(i)^k - (j)^k$ order. Update $T = T \cup \{(i)^k\}$ and $H = H \cup \{(j)^k\}$.

Step 5 (update): Let $k=k+1$. If $k=n+1$ then go to step 6, otherwise go to Step 2.

PHASE 2

Step 6 (initialization): Let $t = 1$ and $m =$ number of groups. Let $(G)^t$ be the setup time matrix between the groups. Let $GT = \emptyset$ and $GH = \emptyset$.

Step 7 (termination): If $m = 1$, a heuristic solution to the scheduling problem is found and stop.

Step 8 (sorting): Sort all the setup times in $(G)^t$ in ascending order. Select the smallest setup time $(g_{ij})^t$ from the sorted list of elements in matrix $(G)^t$. Let $(i)^t$ and $(j)^t$ be the row and column (group) indices, respectively.

Step 9 (elimination):

(a) Eliminate all the setup times in row $(i)^t$ and column $(j)^t$ as well as the setup time $(g_{ij})^t$ from $(G)^t$.

(b) Eliminate all the (group) setup times for element

$(G_{pq})^t \in \{G_{pq} \mid p \in GH \text{ and } q = (i)^t, \text{ or } p = (j)^t \text{ and } q \in GT\}$.

Step 10 (combining groups into larger groups):

(a) If group $(i)^t$ has already been combined with another group in a previous iteration, link group $(j)^t$ to the end of that group containing $(i)^t$, if $(j)^t$ does not already exist earlier in the sequence; update

$GH = GH \setminus \{(i)^t\} \cup \{(j)^t\}$.

(b) If group $(j)^t$ has already been combined with another group in a previous iteration, attach group $(i)^t$ at the beginning of that group containing $(j)^t$, if $(i)^t$ does not already exist later in the sequence; update

$GT = GT \setminus \{(j)^t\} \cup \{(i)^t\}$.

(c) If neither group $(i)^t$ nor group $(j)^t$ have been combined with another group in a previous iteration, combine the two groups in the $(i)^t - (j)^t$ order. Update $GT = GT \cup \{(i)^t\}$ and $GH = GH \cup \{(j)^t\}$.

Step 11 (update): Let $t=t+1$. If $t=m+1$, go to Step 6; otherwise, go to Step 8.

COMPUTATIONAL STUDY

To test the efficiency of the proposed heuristic, we compared HICOM against the general purpose mixed integer linear programming solver CPLEX on random instances with various sizes. The CPU times reported here are from a Dell PowerEdge 2600 with dual Pentium 3.2GHz Xeon processors and 6 GB of RAM. HICOM was coded in Fortran 90 and the mixed integer models were solved by CPLEX 10.2 through the modeling language GAMS.

In the computational study, we used randomly generated instances with the number of jobs ranging from 10 to 120. By the nature of the heuristic HICOM, the algorithm is expected to perform well for scheduling jobs that come in product families. Thus, the description of our random instances uses the following notation: MS = Matrix Size, MM = Matrix Mean, # B = Number of Boxes (i.e., families), MB = Mean of Box, SM = Spread of Matrix, and SB = Spread of Box.

In our simulation we divided the overall changeover time matrix into two. We first consider the setup time for within group changeovers. These changeover times are for changeovers within the box. MB (mean of box) represents the mean setup time of changeovers within the box. For instance, if MB = 7 for the first matrix and MB = 5 for the second matrix, then the mean setup time for the first matrix is more than the mean setup time for the second matrix. SB (spread of box) represents the variation of setup times within the box. For instance, if SB = 3 for the first matrix and SB = 2 for the second matrix, then the first matrix box is more variable than the second matrix box because the spread of the first matrix is larger than the spread of the second matrix. In other words, in a matrix with MB of 7 and SB of 2, the box completion times range from 5 (=7-2) to 9 (=7+2). The box (within a group), as well as between group changeover times, are assumed to be from a uniform distribution. Finally, for a small matrix size, such as 10, the number of boxes (# B) varies from one to three. However for large sample sizes such as 120, the number of boxes ranges from 12 to 40. In general, we allow 3 to 10 jobs per box. Second, we consider the setup time for between group changeovers. These changeover times are labeled MM (matrix mean), and they represent the mean setup time of changeovers between products or items in different groups. By its very nature, $MM > MB$. SM (spread of matrix) represents the variation of setup times between items in different groups. Most of the time we can expect $SM \geq SB$. In the remaining notation, MS represents the matrix size, which ranges from 10 to 120.

Results in the following tables compare HICOM with CPLEX sizes of 50 and 70. When the problem size is higher than 90, CPLEX did not provide incumbent solution for any of the test instances before reaching the CPU limit of 5000 seconds (approximately 1.5 hours). Each table contains 10 instances and provides information on the best objective value, i.e., the total set up time, by HICOM; the objective value by CPLEX, the reported optimality gap from CPLEX, the total CPU time by CPLEX and the gap between HICOM and CPLEX solutions. The Gap is calculated as $(100 \times (\text{HICOM-objective} - \text{CPLEX-objective}) / \text{CPLEX-objective}) \times 100\%$. Finally, the CPU times used by HICOM are not included in any tables because they are essentially zero for all test instances.

Analyzing the tables below, we observe the following. First, as the problem size increases the advantage of HICOM over CPLEX in CPU time increases as well. This is supported by the average CPU times of CPLEX for the 50x50 set is nearly 2,640 seconds, and within specified 5,000 seconds CPLEX could not provide solutions with 1% optimality for any of the 70x70 instances. On the other hand, HICOM used essentially zero seconds for all cases. Second, as the problem size increases the gap between the HICOM and CPLEX solutions decreases. Particularly, the average gaps are 6.5% and 4.6% and the median optimality gaps are 6.5%, and 4.2% for problem sets 50x50 and 70x70, respectively. The CPLEX results were obtained with either an optimality gap of 1% or a time limit of 5000 seconds. Therefore, Tables 1 and 2 suggest that the bigger the problem size is the bigger the advantage of using HICOM.

CONCLUSIONS

The purpose of the heuristic HICOM is to help the schedulers in their day-to-day decision making process by providing an efficient, approximate solution procedure that is fast and flexible. The procedure is more valuable for operations with asymmetric changeover times. In general, as the problem size increases, algorithms with polynomial rates gradually become unusable, whereas algorithms with exponential growth rates rapidly become completely useless. Computational results indicate that the heuristic is very efficient

and reasonably accurate. The performance of the heuristic improves as the clusters become more distinct. The sequencing heuristic should be easily understood by practitioners and is, therefore, more likely to be implemented and accepted by production schedulers. We have identified numerous benefits associated with the application of this procedure. Perhaps the most important benefit of applying this procedure is the potential improvement in efficiency which can lead to substantial savings and improved customer service. Metal processing, textile, plastics, paint, commercial printing can potentially benefit from successful application of this heuristic.

Table 1
Results for 50x50 Instances

Instances	HICOM-obj	CPLEX-obj	CPLEX-gap	CPLEX-cpu time(in secs.)	Gap
1	130.2318	128.671	1.0%	347.81	1.2%
2	138.9405	130.042	1.0%	73.86	6.8%
3	121.811	115.702	1.0%	4,769.89	5.3%
4	107.696	103.255	8.9%	5,000.22	4.3%
5	141.2336	128.692	1.0%	169.65	9.7%
6	91.09521	81.384	9.8%	5,000.07	11.9%
7	125.0127	117.862	1.0%	670.66	6.1%
8	129.5313	120.762	1.0%	363.56	7.3%
9	105.0497	103.886	11.7%	5,000.25	1.1%
10	124.041	111.379	3.1%	5,000.07	11.4%
Average	121.4643	114.1635	4.0%	2,639.60	6.5%
Median	124.5269	116.782	1.0%	2,720.28	6.5%
Max	141.2336	130.042	11.7%	5,000.25	11.9%

Table 2
Results for 70x70 Instances

Instances	HICOM-obj	CPLEX-obj	CPLEX-gap	CPLEX-cpu time(in secs.)	Gap
1	183.2023	178.47	6.5%	5000.16	2.7%
2	159.8245	157.736	9.6%	5000.15	1.3%
3	133.8276	121.302	10.5%	5000.12	10.3%
4	162.8845	167.804	9.8%	5000.18	-2.9%
5	193.4104	181.911	3.2%	5000.28	6.3%
6	185.9064	177.744	2.4%	5000.15	4.6%
7	180.849	174.591	1.5%	5000.24	3.6%
8	188.0851	174.102	2.0%	5000.13	8.0%
9	147.0303	135.162	11.2%	5000.12	8.8%
10	170.5258	164.346	5.9%	5000.18	3.8%
Average	170.5546	163.3168	6.3%	5000.171	4.6%
Median	175.6874	170.953	6.2%	5000.155	4.2%
Max	193.4104	181.911	11.2%	5000.28	10.3%

References are furnished from the first author upon request.