Beam search for sequencing point operations in flat plate manufacturing

Shayan, E.
IRIS, Industrial Engineering Group,
Swinburne University of Technology
Hawthorn Vic 3122 Australia

Al-Hakim, Latif
The University of Southern Queensland
Toowoomba Qld 4350 Australia

Abstract

This paper considers point operations problems in which several operations need to be performed at pre-specified locations on a flat plate, by a CNC machine. This paper develops a new procedure based on previous termbeamnext term search methodology. The proposed method was tested using real data. The results show that the proposed method provides superior results over the current procedure used by an Australian company. The method significantly reduces the non-value added (travel+changeover) time.

1. Introduction

Point operations in flat plate manufacturing are concerned with conduct of manufacturing operations at a set of distinct points, $P = \{1, 2, ..., N\}$ distributed over a flat surface at known locations. Each point, $i \in P$, is centred at a location (x_i, y_i) and comprises a specific set O_i of m_i operations, $O_i = \{O_{i1}, O_{i2}, ..., O_{im_i}\}$, which have to be processed according to a given technical precedence order. Each operation requires a specific tool. The processing time may vary from one point to another.

As an example of point operations, consider a CNC machine performing a variety of operations at several locations on a flat plate such as marking, drilling, boring, and counter-sinking. Each operation may need a different tool to be loaded which takes a known time. After some point operations, the immediate area is washed with a high-pressure water jet in order to clear the swarf. The objective of the

point operations problem is chosen to be the determination of an optimal sequence for point operations subject to the following constraints:

1. *Point-operation sequence constraint.* The operations at each point should be performed at the specified sequence using the specified tool for each operation.
2. *Heat constraint.* After each point operation, the area should be cooled down before performing another operation at the same location. In practice, a problem may occur if this constraint is violated.
3. *Plate washing constraint.* After some point operations, the immediate area is washed with a high-pressure water jet in order to clear the swarf. Usually, the swarf is pushed away to lie in a roughly semi-circle area at a particular angle to the location of that point operation.

According to constraints 2 and 3, it is unlikely that two successive operations can be performed at the same location without the second operation being delaying until cooling or washing of the area after the first operation.

The total time required to perform all operations comprises operation times, head travelling time, tool change time (setup time), and washing-up times. All the time elements except the operation time are non-value added. The objective is to determine the route with minimum time to process all the operations required. However, the processing time of each operation is fixed and independent from the route. Accordingly, the objective of the problem can be stated as to find a route with minimal non-value added time.

The problem can be viewed in terms of a network (N, E), where $\{N\}$ is the set of locations and $E = \{e_{ij}\}$ represents non-value added times (cost) associated with movement between the locations i and j. In the case where there is only one operation at each location or all the point operations at a location can be performed sequentially, then the problem can be formulated as a classical travelling salesman problem (TSP) in which there are N points (cities) that must be visited and the objective is to minimise the total distance (time) travelled (Pinedo, 1995), that is to minimise $\sum e_{ij}$, for $i,j \in N$. However, this is not the case and there are three significant differences that make many heuristics and procedures developed to solve classical TSP fail to solve the point operations problem, motivating us to find other techniques:

1. Normally, there is multiple operations (or visits) needed to be performed at each location, while TSP usually requires each city to be visited once.
2. For the purpose of TSP, the multiple operations at the same location may be considered as different cities. However, the multiple operations at each location should be performed according to a specified order. In TSP, there is no prior determination of visiting cities.
3. Unlike TSP, point operations problem does not require to finish at the starting point by closing the loop.

2. Relevant algorithms

Point operations problem is within the class of discrete optimisation problems in which, given a finite set of solutions S and an objective function F, one has to find a solution $s^* \in S$, such that $F(s^*) \preceq F(s)$ for all $s \in S$.

Local search techniques are widely used to solve discrete optimisation problems (Morton & Pentico, 1993; Pinedo, 1995). Local search is an iterative procedure that selects a solution $s \in S$ and defines its

neighbourhood set. For each $s \in S$, the neighbourhood set contains a subset of solutions that can be reached in one step by moving from s. The procedure will terminate with some solution s^* after pursuing a predefined stopping rule. In general, local search techniques have the disadvantage that it is possible to get back to a solution already visited. Therefore oscillation around local minima is possible. This may lead to a situation where much computational time is spent on a small part of the solution space. Glover (1989, 1990) describes a technique referred to as 'Tabu Search' to avoid such oscillation. Glover suggests storing all the visited solutions in a list called Tabu list and accepting only the new solutions not contained in the list. However, storing all visited solutions and testing if a candidate solution belongs to the list is generally too expensive, both in terms of memory and computational time.

Branch-and-bound is another methodology to solve combinatorial problems. It is based on the idea of intelligently enumerating all feasible solutions (Brucker, Jurisch, & Sievers, 1994). Of course, this will also be too expensive except for small problems. The basic idea of the methodology is to conceptualise the problem as a decision tree, which comprises internal nodes and leaf nodes. An internal node represents a partially completed solution from which a number of new branches grow. These in turn become new internal nodes for branching again, and so on. If the lower bound of the partial solution for a node is greater than or equal to an upper bound calculated for the problem, then this node cannot yield a better solution for the problem. Thus we need not continue to branch from the corresponding node in the branching tree. This node becomes a leaf node. A leaf node represents a complete solution (or dead end) which cannot be branched out any further. The leaf node with the smallest upper bound represents the best-selected solution.

Beam search (Lowerre, 1976; Ow & Morton, 1988) is a tree search technique with breadth-first strategy, but unlike the branch-and-bound methodology, no backtracking is allowed (Kadipasaoglo, Arostegui, Zalila, Peixoto, & Khumawala, 1998; Morton & Pentico, 1993). Instead of branching the whole tree and waiting to throw away parts of the tree that are guaranteed useless, we may throw away parts of the tree that are likely to be useless. One important issue here is to have a good measure of what 'likely' means; another is to throw away tree's parts that save a lot of effort without taking much risk. Beam search methods have been used to develop computer chess programs (Anthony & Schaffer, 1990). They employ a mixture of local search 'intensification' and branch-and-bound 'diversification' strategies (Morton & Pentico, 1993). Section 3 provides some insights on beam search.

3. Beam search

The following terminology explains the beam search decision tree within the context of point operations problem. We use the same terminology of Chang, Matsuo, and Sullivan (1989) with some adaptation to suit the point operations problem.

Node:	a partial solution which specifies the route time required to process subset of point operations using specified tool.
Root node:	node zero which corresponds to the null set.
Level:	depth of a node from a root node.
Beam node:	a node from which other nodes may be created.
Beam width:	the maximum number of bean nodes allowed at each level, β.
Branch factor:	the maximum number of nodes allowed from a beam node, α.

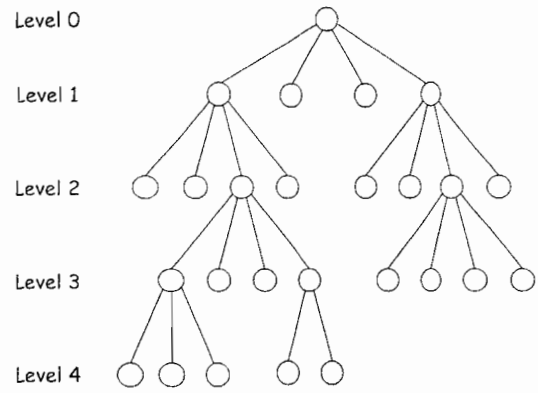

Fig. 1. A four levels beam search decision tree with $\beta = 2$ and $\alpha = 4$.

At each level of the search, the beam search is limited to the most promising β nodes. Each node of β represents a partial solution using a specified tool to perform all the point operations that can be processed simultaneously by the same tool. From each beam node of β another subset of point operations may be performed. The function used to evaluate internal nodes and to select the likelihood of a node to be a beam node is called the evaluation function. One variation is called filtered beam search, in which only certain number of moves (point operations) at each beam node passing a crude preliminary filter are actually explored. The evaluation function may consider only the effect of a given partial solution by looking only to certain number of operations ahead rather than to consider the effect of all the remaining operations (Ow & Morton, 1988).

Increasing the beam width β and the branch factor α lead to better solutions but at the expense of increasing computational effort. A good balance between the computational time and quality of solution could be achieved by modifying the beam width and the branch factor. Fig. 1 gives an example of beam search decision tree with $\beta = 2$ and $\alpha = 4$.

De, Ghosh, and Wells (1992) used beam search to examine the single machine scheduling problem and concluded that beam search was very effective. Experimental results reported by Kadipasaoglo et al. (1998) shows that beam search compared favourably against widely used heuristics existing in the literature.

4. Procedure

The procedure requires the following input: N, (x_i, y_i) and O_i for each point $i \in P$. For each operation in O_i, the procedure demands the tool number and processing time. The procedure requires values for β and α. The sequence of operations at each $i \in P$ is stored in a list called LIST. Information such as tool setup time and area washing time for various CNC machines is formulated as a subroutine attached to the procedures. The procedure has a subroutine to determine the tool path based on TSP algorithm.

The procedure considers the first operation for each point in LIST and determines the tools required to perform these operations. Only α tools which can perform the highest number of operations are selected for branching. For each branch TSP subroutine is applied by considering only those operations that can

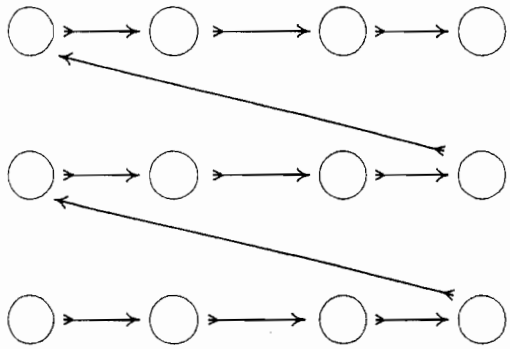

Fig. 2. The x-axis zigzag point operations.

be performed by the selected tool. The cost (non-value added time) for each node is determined. We arrange the nodes in ascending order according to their cost values. The first β branches are selected as candidates for branching in the next level. All other nodes are thrown away. LIST is updated by removing from it all the processed operations. The procedure is repeated until LIST is empty. The lowest cost node among the last formed nodes forms the best solution to the problem. Given m as the maximum number of operations at any given location, the complexity of the problem is $O(b(mN)^2)$, thus it becomes possible to produce a good solution for large problem at a reasonable computational time.

5. Case study

An Australian company designs and manufactures profile cutting machines. Two of their CNC machines are capable of performing a variety of point operations. The company aims at developing an algorithm to solve medium size point operations problems in under 10 s.

The current procedure used by the company produces a sequence based on an x-axis zigzag path (Fig. 2). The procedure cycles through the point operations considering one tool at a time in a predetermined zigzag path. The tool moves along the x-axis before jumping to the next level of point operations. The arrows in Fig. 2 represent the path. The current procedure can be easily modified to produce a sequence based on y-axis.

It is readily known that the current procedure gives good results when the points have the same sequence of operations, require the same tools, are closely spaced and are in a grid-like pattern. Otherwise, poor solutions are generated.

6. Computational results

The developed procedure has been implemented on C language under the Linux operating system using GNU g++ version 2.7.2.1-2.

The proposed procedure was tested using a range of test problems. Some of these problems have been provided by the company. The size of these problems range from 20 to 300 points with up to five operations at each point. For all the problems we consider the beam width β and the branch factor α

Table 1
Comparisons between the current and developed procedure

Problem	Size N	Non-value added time (s)		Improvement (%)
		x-Axis zigzag	Beam search	
1	20	281.70	109.40	61.16
2	24	306.10	189.23	38.20
3	24	298.00	207.9	30.23
4	48	6298.00	2134.56	66.10
5	50	283.13	109.45	61.34
6	100	2580.9	1105.50	57.17
7	100	3998.80	2905.80	27.33
8	100	4978.10	1787.80	64.41
9	230	2245.47	1112.30	50.55
10	300	6512.80	2259.28	65.31

equal to four. All the problems are solved with CPU time less than 10 s. Table 1 compares the non-value added time (travel + tool changeover) required by the CNC machine to implement the task using the current procedure and the developed one.

Table 1 shows that the developed procedure produces superior results over the currently used procedure. The reduction in the non-value added time ranges between 27 and 66%. The developed procedure provides significant reduction in non-value added time where the distribution of the points is not uniform as in test problem 1 and 8, or there is a high rate of tool exchange as in problem 10 or both as in problem 4.

7. Conclusions

The proposed method has been tested on several real data sets and is capable of solving practical size problems within a reasonable computational time. The proposed method produces superior results over the current zigzag procedure. The method contributes up to 66% reduction in non-value added time. More extensive testing would be needed using different beam width and more complex evaluation function for better conclusion on beam search.

Acknowledgements

We acknowledge the support of Farley Cutting Systems, Pty, Australia, for providing the practical problems for this study. We would also like to thank an anonymous referee for his/her constructive comments.

References

Anthony, M. T., & Schaffer, J. (1990). *Computers, chess and cognition*, New York: Springer.

Brucker, P., Jurisch, B., & Sievers, B. (1994). A branch and bound algorithm for the job-shop scheduling problem. *Discussion of Applied Mathematics*, *40*, 107–127.

Chang, Y., Matsuo, H., & Sullivan, R. (1989). A bottleneck-based beam search for job scheduling in a flexible manufacturing system. *International Journal of Production Research*, *27*, 1949–1961.

De, P., Ghosh, J., & Wells, C. (1992). Heuristic estimation of the efficient frontier for a bi-criteria scheduling problem. *Decision Sciences*, *23*, 596–609.

Glover, F. (1989). Tabu search—Part 1. *ORSA Journal of Comuting*, *1*, 190–206.

Glover, F. (1990). Tabu search—Part 2. *ORSA Journal of Comuting*, 2, 4–32.

Kadipasaoglo, S.N., Arostegui, M., Zalila, F., Peixoto, J., & Khumawala, B. (1998). Job-shop scheduling: Application of Tabu search, simulated annealing, genetic algorithms and beam search. *Proceedings of the Sixth International Workshop of Project Management and Scheduling*, Bogazici University, Istanbul, Turkey.

Lowerre, B.T. (1976). *The HAPPY Speech Recognition System*. PhD Dissertation. Carnegie-Mellon University.

Morton, T. E., & Pentico, D. W. (1993). *Heuristic scheduling systems*, New York: Wiley.

Ow, P. S., & Morton, T. E. (1988). Filtered beam search in scheduling. *International Journal of Production Research*, *26*, 35–62.

Pinedo, M. (1995). *Scheduling: theory, algorithms, and systems*, New Jersey: Prentice-Hall.