An intelligent methodology for optimising machining operation sequence by ant system algorithm

Sneha Singh and Sankha Deb*

Department of Mechanical Engineering,
Indian Institute of Technology Kharagpur,
Kharagpur-721302, India
E-mail: snehasingh.iitkgp@gmail.com
E-mail: sankha.deb@mech.iitkgp.ernet.in
*Corresponding author

Abstract: The paper describes an intelligent ant system-based algorithm for automatic generation of optimal sequence of machining operations required to produce a part, based on minimising the number of tool changes and set-up changes subject to satisfying all precedence constraints during manufacturing. The MATLAB programme for the algorithm uses a list of machining operations, tool approach directions, and the precedence constraints between the operations as inputs. It generates only feasible sequences of operations and finds out an optimal sequence among them. The concept of specific selection of a starting node at the beginning of each ant cycle and introducing a precedence check in the transition rules reduces the computation time significantly. A comparative study shows that for a demonstration run, the proposed ant system-based approach performed faster than previously developed methodologies for ant colony optimisation as well as a genetic algorithm-based optimisation techniques.

Keywords: operation sequence; optimisation; ant system; computer-aided process planning; CAPP; ant colony optimisation; ACO.

Reference to this paper should be made as follows: Singh, S. and Deb, S. (2014) ‘An intelligent methodology for optimising machining operation sequence by ant system algorithm’, *Int. J. Industrial and Systems Engineering*, Vol. 16, No. 4, pp.451–471.

Biographical notes: Sneha Singh obtained her Bachelor’s of Technology in Manufacturing Science and Engineering from Indian Institute of Technology Kharagpur in the year 2011. She is currently pursuing her PhD in the University of Warwick in the area of electric vehicles’ sound quality engineering. Her research interests include computer-based manufacturing technologies, production planning, product perception and structured evaluations.

Sankha Deb obtained his PhD in Industrial Engineering from Ecole Polytechnique Montreal, Canada and Masters in Manufacturing Process Engineering from Indian Institute of Technology Kharagpur. He served as an Assistant Professor at Indian Statistical Institute Calcutta and at Indian Institute of Technology Guwahati. He was invited as a Visiting Professor in University of Montreal. Currently, he is an Assistant Professor of Mechanical Engineering at Indian Institute of Technology Kharagpur. He has published many research papers and co-authored one book on robotics technology and flexible automation. His research interests are computer integrated manufacturing, automation and robotics, and soft computing.

1 Introduction

Computer-aided process planning (CAPP) is the use of computer technology to aid in the process planning of a part in manufacturing. It is a crucial link between computer-aided design (CAD) and computer-aided manufacturing (CAM). It involves taking the part design specifications as input, interpreting the design data to identify the machining features present in the part design, and selecting operations required to manufacture the part, machines, cutting tools, the set-ups, sequence of the machining operations, cutting parameters, etc. The output is the process plan containing detailed steps of manufacturing.

Operation sequencing, i.e., selection of the sequence in which machining operations are to be performed, is a crucial part in process planning and has gained much attention (Xu et al., 2011; Singh and Singh, 2012). A good process plan for a part depends on the selection of optimum sequence of the operations to manufacture the part (Salehi and Bahreininejad, 2011). One of the major challenges for optimisation here lies in minimising the overall manufacturing cost. Several researchers have approached the problem of minimising the cost by minimising changes in parameters like machine changes, tool changes, and set-up changes (Ma et al., 2000; Krishna and Rao, 2006; Kumar and Deb, 2012). We have adopted a similar approach by assuming that, in a particular industry and while operating a particular machine, the manufacturing cost increases with increase in the machining time for a part, provided other factors like machine set-up time, number of operators, etc., are held constant. Based on this assumption, manufacturing cost can be minimised by minimising the machining time of a part. In order to manufacture a part, the same set of operations when performed in a certain sequence may require much less machining time owing to a smaller number of inter-operation cutting tool changes and machine set-up changes. Therefore, problem of minimising cost/time may be considered equivalent to minimising number of set-up changes and tool changes while manufacturing a part. However, while deciding the operation sequence, the precedence constraints must not be violated.

In this paper, an ant system-based algorithm has been adopted to automatically generate an optimal operation sequence to minimise the manufacturing cost by minimising the number of tool changes and the number of set-up changes subject to various precedence constraints. A programme implementing the algorithm in MATLAB 7.10 takes input files containing the list of features, their machining operations, tool approach directions and the precedence constraints among the various machining operations. The programme uses ant system proposed by Dorigo et al. (1996) as a search heuristic and provides the best known sequence from all the feasible sequences of machining operations generated by the algorithm. In this paper, we use the term ‘optimal’ for the best found solution during the run of the programme. Thus, here an optimal sequence refers to the best found sequence among the feasible sequences generated by the algorithm. However, in practice there may be more than one optimal solution for a part and situations may arise when another feasible sequence with a cost lower than or equal to the one found by the algorithm exists, but such instances are very unlikely.

Krishna and Rao (2006) have also addressed a similar problem of operation sequencing in CAPP using an ant colony-based algorithm. The paper presents a critical analysis and performance comparison of both the algorithms which shows that the algorithm presented by us performs much faster and better.

The rest of the paper is organised as follows: Section 2 describes previous research works in operation sequence optimisation using evolutionary computation techniques. Section 3 gives a brief description of ant system proposed by Dorigo et al. (1996) and some of its popular variations. Section 4 describes our approach and algorithm to solve the problem at hand. Section 5 presents the results and discussions of implementing our algorithm on two parts taken as examples for illustration as well as a comparative study with algorithm by Krishna and Rao (2006) and an equivalent genetic algorithm (GA) approach. Section 6 presents the conclusions and future recommendations.

2 Review of previous research

Several researchers have used evolutionary computation algorithms for solving optimisation problems related to various aspects of process planning such as operation sequence optimisation, set-up planning, process parameter optimisation, etc. as well as various aspects of production planning and control such as scheduling, facility layout planning, etc. A summary of the previous research work is presented below.

Since Goldberg (1989) first proposed the idea to use GAs for optimisation, many researchers have used GA-based approach for operation sequence optimisation but each had their own optimisation criteria and constraints. For example, Dereli and Filiz (1999) used the criteria of minimising the number of tool changes and tool travel distances; Reddy et al. (1999) used minimisation of machining parameter change, tool change, set-up change, and machine change; while Li et al. (2005) used a fitness function for optimisation based on minimising processing time and production cost. Further, Hua et al. (2007) used the criterion of minimising the machining cost, while Salehi and Tavakkoli-Moghaddam (2009) extended their optimisation criteria to include minimisation of machine cost, tool cost, machine change cost, tool change cost, set-up change cost, etc. The choice of optimisation criteria is crucial and depends on the problem at hand. It is equally important to include constraints that must be satisfied in order to machine a particular part. Choice of constraints again depends on the problem and the availability of machining data. For example, Qiao et al. (2000) used various process planning rules including precedence rules, clustering rules, adjacent order rules, and so on; Kumar and Deb (2012) have attempted to optimise operation sequences in set-up planning based on minimising the number of set-up changes and tool changes, subject to various machining precedence constraints. GAs have also been used for solving optimisation problems in production planning and control such as process route optimisation (Bo et al., 2006), multi-objective scheduling (Azadeh et al., 2010) and facility layout planning (Kundu and Dan, 2012).

More recently, other evolutionary computation algorithms like particle swarm optimisation (PSO), simulated annealing (SA), tabu search (TS) and ant colony optimisation (ACO) have also gained popularity and have been applied to solve the problems of optimisation in process planning and production planning and control. PSO methods have been explored by Guo et al. (2006) for operation sequence optimisation based on minimisation of machining time, number of machine changes, set-up changes, tool changes and the set-up time. Recently, Rao et al. (2010) have used artificial bee colony, harmony search, and PSO algorithms to give an optimal combination of process variables of ultrasonic machining (USM) process. The optimisation criteria used is maximisation of material removal rate in USM subject to constraints of minimum surface roughness. PSO has also been used Pawar et al. (2010) to find the optimal combination of grinding process parameters (wheel speed, work-piece speed, depth of dressing and lead of dressing) with multiple objectives of minimisation of production cost, maximisation of production rate and minimisation of surface roughness subject to constraints of thermal damage, wheel wear, and machine tool stiffness. Ma et al. (2000) have used SA-based approach for operation sequencing optimisation based on minimisation of tool change and set-up change costs. Rahim and Shakil (2011) recently used an integrated model of economic production quantity to obtain total expected cost of a single-item manufacturing process in terms of production planning, quality control, and preventive maintenance, in which they used a TS algorithm to find the optimal values of quality control parameters, for different preventive maintenance levels, based on minimising the expected total cost and quality control cost.

ACO techniques with appropriate variations have also been used for solving optimisation problems in process planning and production planning and control. Krishna and Rao (2006) proposed ant colony system to solve operation sequence optimisation based on minimisation of machine change, set-up change, tool change, and machining parameter change. Although the algorithm performed better than other algorithms like GA, it has only been applied to simple problems with less than 20 operations. Researchers have also studied variations of ACO and in some cases found them to perform better than the original ant system algorithm. For example, Gagne et al. (2002) have introduced a look-ahead information (about the potential of current partial solution) in the transition rules that has been found to perform better on some of the single machine scheduling problems with sequence dependent set-up times. However, significant increase in the computation time is one of the shortcomings faced by the algorithm. Iredi et al. (2001) have used multi-colony ant algorithm for a single machine total tardiness problem with changeover costs, but a non-dominant solution front is found only after 300 generations which makes it very time consuming for complex problems. It has been found from the literature review that the ACO methodologies perform comparatively better than some other evolutionary algorithms but ACO algorithms employ random search thus, even if the convergence of solution space is guaranteed, the convergence rate may decrease with increase in the problem complexity. Therefore, research is required to further enhance the application ant colony-based methodology.

Some researchers have used combination of different evolutionary computation approaches. For example, Li et al. (2002) have used both GA and SA for operation sequence optimisation, Hua et al. (2007) have used fuzzy logic-neural network for determining fitness criteria and GA for final optimisation. Rameshkumar et al. (2011) have attempted to solve the permutation flowshop scheduling problem, with the objective of minimising the completion-time variance of jobs, using PSO algorithm that is inspired from the solution construction procedures that are used in ACO algorithms. Azizi et al. (2009) proposed an algorithm for flow shop scheduling where SA acts as the main driver and has two short-term memories (to simultaneously prevent cycling and to further explore the area surrounding good solutions) and an evolution-based long term memory for diversification used together with a GA crossover operator. Another hybrid algorithm based on SA and TS is recently employed to improve the solution quality of the complex multi-depot vehicle routing problem which uses extended rules to accommodate large-scale orders by splitting the delivery (Dharmapriya et al., 2012).

3 Ant system

The ants are almost blind but can still find out the shortest route from their colony to the feeding sources and back. The research work revealed that the medium used to communicate information among individual ants regarding paths and used to decide where to go consists of pheromone trails (Dorigo et al., 1996). A moving ant lays some pheromone in varying quantities on the ground, thus marking the path by a trail of this substance. While an isolated ant moves essentially at random, but on encountering a previously laid trail can detect it and decide with high probability to follow it, thus reinforcing the trail with its own pheromone. The collective behaviour that emerges is a form of autocatalytic behaviour where the more the ants following a trail, the more attractive that trail becomes for being followed. The process is thus, characterised by a positive feedback loop, where the probability with which an ant chooses a path increases with the number of ants that previously chose the same path (Dorigo et al., 1996).

The above mentioned properties of ants have resulted in the development of many ACO algorithms to provide an effective search technique for combinatorial optimisation problems. The approach put forward in the paper considers the ant system proposed by Dorigo et al. (1996), one of the earliest algorithms in the family of ACO. This algorithm incorporates the characteristics behind the ant's behaviour in finding the shortest path from a source to a destination. However, in this algorithm the artificial 'ants' have some major differences with a real (natural) ones. They have some memory, they are not completely blind, and they live in an environment where time is discrete.

Due to its flexibility of application into a variety of optimisation problems many popular variations of ACO algorithms have emerged. In Elitist ant system, at every cycle the ant having the best tour deposits additional pheromone thereby increasing the value of the trail on edges belonging to its tour (Dorigo et al., 1996). This is to ensure that the best solution is not lost. The max-min ant system (MMAS) adds maximum and minimum pheromone amounts and restricts the pheromone deposit by only global best or iteration best tour. All edges are initialised to a maximum pheromone amount at the start of the algorithm and then reinitialised to the maximum pheromone amount when approaching stagnation (Stützle and Hoos, 2000). The ant colony system is developed by Dorigo and Gambardella (1997) with the view of application to the travelling salesman problem (TSP) and the asymmetric travelling salesman problem (ATSP). It differs from the ant system in that it uses a state transition rule and local pheromone updating rule to guide the search. The global updating rule (as used in the original ant system) is only applied to the edges belonging to the best ant tour. In the rank-based ant system by Bullnheimer et al. (1999) all solutions are ranked according to the length of the path covered in each solution. The amount of pheromone deposited is in inverse proportion to the length. The most recent addition to the ACO family, the hyper cube ACO by Blum and Dorigo (2004) limits the pheromone values to the interval [0, 1] by introducing changes in the pheromone update rule.

4 Proposed methodology for optimisation of machining operation sequence by ant system

The approach used to solve the problem at hand is analogous to solving a TSP but with some differences. This approach is discussed in Section 4.1. An algorithm is then developed based on the discussed approach to solve the problem at hand while using ant system as a search heuristics. The algorithm is further improved to ensure that it generates operation sequences which satisfy all precedence constraints. Section 4.2 discusses the algorithm in detail.

4.1 Problem description

The problem of finding an optimal sequence of, machining operations required to manufacture a part, such that the total manufacturing cost is minimised and all precedence constraints are satisfied is analogous to the well-known TSP. The analogy of the problem at hand with the TSP is as follows:

- towns correspond to operations that must be performed to make a part feature
- distance travelled in going from one town to the other corresponds to the cost of performing the second operation after the first
- a tour corresponds to a sequence of operations
- optimal tour corresponds to the sequence of operations that has minimum total cost.

As already discussed in the introduction our rationale is that cost can be minimised by minimising the number of cutting tool and set-up changes for a single machine environment. To implement this we introduce four matrices namely, cost matrix, tool change matrix, set-up change matrix, and default matrix. Every element of the cost matrix gives the value of cost incurred in performing an operation (corresponding to the column number) after another (corresponding to the row number). Let us call the process of selecting or performing one operation after another an 'operation transition' or simply 'transition'. Then each element of the cost matrix is calculated as a weighted sum of tool change cost, set-up change cost and a default cost corresponding to that transition. A set-up change takes more time than a tool change hence the set-up change is given a higher weightage or 'cost factor' when calculating the cost for an operation transition [see equation (4)].

For demonstration we have chosen a default cost factor of 5 units for performing an operation transition, an additional tool change cost factor of 40 units if the transition is accompanied by a tool change, and an additional set-up change cost factor of 100 units if it is accompanied by a set-up change. It is important to note here that if no default cost is assigned to perform a transition, then minimum total cost would be zero for a part that can be produced with no set-up and no tool change [see equation (4)]. In such a case the presented algorithm would face a divide by zero error [see equation (7) and Section 4.2]; therefore a non-zero default cost is required. We multiply the cost factor by 1 or 0 depending upon whether or not there is a change in the corresponding parameter. The total cost in performing a transition is the sum of tool changes and set-up changes (1: if there is a change; 0: if there is no change) multiplied with their corresponding cost factors plus a default cost.

Let D, C, T and S denote the default, the cost, the tool change and the set-up change matrices respectively. Also let d_{ij}, t_{ij} and s_{ij} denote the $(i, j)^{\text{th}}$ element of the matrices D, T and S respectively. The algorithm generates the tool change, the set-up change and the default matrices directly from the input data and uses it to calculate the cost matrix using equations (1) to (4).

$$d_{ij} = \{1; \quad \text{for all } i \text{ and } j\} \tag{1}$$

$$t_{ij} = \begin{cases} 1; & \text{if operation } j \text{ needs different tool than operation } i \\ 0; & \text{otherwise} \end{cases} \tag{2}$$

$$s_{ij} = \begin{cases} 1; & \text{if operation } j \text{ needs different setup than operation } i \\ 0; & \text{otherwise} \end{cases} \tag{3}$$

$$C = 5 \cdot D + 40 \cdot T + 100 \cdot S \tag{4}$$

Using equations (1) to (4), if we have two operations *A* and *B* and let *B* requires the same cutting tool but a different set-up than *A*. So the cost in performing *B* after *A* would be 5 + 40 × 0 + 100 × 1, i.e., 105 units. So, the (*A*, *B*)th element of the cost matrix will have the value 105. Thus, the problem at hand reduces to taking list of machining operations, tool approach directions and the precedence constraints as inputs. Calculating the matrices *D*, *T*, *S*, and *C*, and using the ant system as a metaheuristic to generate sequences and minimise the total cost of the sequences generated. From the inputs, a tool change and a set-up change is determined by a difference in the type of operation and the tool approach direction respectively. The total cost of a sequence is obtained by adding the cost elements of every transition in the sequence, without adding the cost of returning to the starting element (unlike a TSP). For example, for a sequence 1 – 4 – 3 – 2, total cost $C = c_{14} + c_{43} + c_{32}$ where c_{ij} is the (*i*, *j*)th element of the cost matrix *C*, i.e., the cost of performing operation *j* after *i*.

4.2 Algorithm

The ant system by Dorigo et al. (1996) is adapted to suit the problem at hand and is described as follows:

Let *n* be the total number of operations required to machine the part and *N* be the set of *n* operations. Also, let *m* be the total number of ants. $\tau_{ij}(t)$ represents the intensity of trail on the path of an ant in (*i*, *j*)th operation transition at time *t*. Thus, it indicates how many ants in the past have chosen the transition (*i*, *j*). Each ant at time *t* chooses the next operation, where it will be at time *t* + 1. Therefore, if we call an iteration of the algorithm the *m* moves carried out by the *m* ants in the interval (*t*, *t* + 1), then after every *n* iterations of the algorithm or a 'cycle' each ant has completed an operation sequence. At this point, the trail intensity is updated according to equations (5) to (6) adapted from Dorigo et al. (1996). Here, ρ is a coefficient such that $(1 - \rho)$ represents the evaporation of trail between time *t* and *t* + *n*.

$$\tau_{ij}(t+n) = \rho.\tau_{ij}(t) + \Delta\tau_{ij} \tag{5}$$

$$\Delta\tau_{ij} = \sum_{k=1}^{m} \Delta\tau_{ij}^{k} \tag{6}$$

Equation (7) [adapted from Dorigo et al. (1996)] gives $\Delta\tau_{ij}^{k}$, i.e., the quantity of trail substance per unit cost (pheromone in real ants) for (*i*, *j*)th operation transition by the *k*th ant between time *t* and *t* + *n*. Here, *Q* is a constant and C_k is the total cost for the operation sequence of the *k*th ant.

$$\Delta\tau_{ij}^{k} = \begin{cases} Q/C_k; & \text{if ant } k \text{ makes } (i,j)^{\text{th}} \text{ transition in its sequence between time } t \text{ and } t+n \\ 0; & \text{otherwise} \end{cases} \tag{7}$$

The coefficient ρ must be set to a value greater than 1 to avoid unlimited accumulation of trail. $\Delta\tau_{ij}(0)$ is set to a small positive constant c. In order to have a feasible sequence an ant must choose all n operations exactly once. Thus, each ant is associated with a data structure called the tabu list that saves the operations that are already a part of the current sequence up to time t and, forbids the ant to choose them again before n iterations (a sequence) is complete. When a sequence is complete, the tabu list is used to compute the ant's current sequence and the total cost of that sequence. The tabu list is then emptied and the ant is free again to choose. $TABU_k$ is a dynamically growing vector which contains the tabu list of the kth ant, $tabu_k$ is the set obtained from elements of $TABU_k$ and $tabu_k(s)$, the sth element of the list (sth operation visited by kth ant in the current sequence).

The visibility η_{ij} is the quantity $1 / c_{ij}$, where c_{ij} is the cost incurred in going from operation i to j. It emphasises that the lesser the cost incurred the more desirable that operation is. This quantity is not modified during the run of the algorithm. The transition probability from operation i to j for the kth ant as adapted from Dorigo et al. (1996), is:

$$p_{ij}^{k}(t) = \begin{cases} \dfrac{[\tau_{ij}(t)]^{\alpha} \cdot [\eta_{ij}]^{\beta}}{\sum\limits_{k \in allowed_k} [\tau_{ik}(t)]^{\alpha} \cdot [\eta_{ik}]^{\beta}}; & \text{if } j \in allowed_k \\ 0; & \text{otherwise} \end{cases} \tag{8}$$

where $allowed_k = \{N - tabu_k\}$ and α and β are parameters that control the relative importance of trail versus visibility. Hence, transition probability is a trade-off between visibility (which says that operation transitions with less cost should be chosen with high probability, thus implementing a greedy constructive heuristic) and trail intensity at time t [that says that if a lot of ants have chosen transition (i, j) in the previous cycles, then it is highly desirable; thus implementing the autocatalytic process].

At time zero an initialisation phase takes place during which ants select a starting operation and initial values $\tau_{ij}(0)$ is associated with each transition. The first element of each ant's tabu list is set to be equal to its starting operation. Thereafter, every ant chooses to make (i, j)th transition with a probability given by equation (8) from Dorigo et al. (1996). After n iterations all ants have completed a sequence, and their tabu lists will be full; at this point for each ant k the value of C_k is computed and the $\tau_{ij}(t)$ values are updated according to equations (5) to (7).

The optimal sequence for the cycle, found by the ants (i.e., min C_k, $k = 1,\ldots,m$) is saved and all tabu lists are emptied. This process is iterated until the cycles' counter reaches the maximum (user-defined) number of cycles NC_{MAX}, or all ants generate the same sequence (i.e., when the average node branching (ANB) becomes less than a pre-defined value $ANB_{\min}$). The latter is stagnation behaviour because it denotes a situation in which the algorithm stops searching for alternative solutions.

A unique feature of our approach to solve the problem at hand is the implementation of a separate algorithm to read the precedence data into a precedence matrix P given by equation (9) where p_{ij} denotes the (i, j)th element of P. When calculating the probability of performing operation *j* after *i*, all operations that must precede *j* are stored in a separate row vector *Pre_j*. This is done by extracting the row numbers of all the elements in *j*th column with a value of 1. From *Pre_j* we check if all its elements are already present in the performed operations or not. If at least one of the elements is missing (means there is at least one operation that must be performed before *j* and has not yet been performed) then the probability of going to operation *j* becomes zero. This ensures that every sequence generated is a feasible one.

$$p_{ij} = \begin{Bmatrix} 1; & \text{if operation } i \text{ must precede } j \\ 0; & \text{otherwise} \end{Bmatrix} \tag{9}$$

In order to avoid an infeasible sequence, it is equally important that an ant starts with an operation that has no precedence constraint. Thus, another innovation in our algorithm is the use of vector *Starts* that stores the operations numbers which have no precedence constraints. This is done by extracting the row numbers of all the rows in *P* in which every element is zero. At the first iteration of every cycle, the algorithm places an ant in a starting operation randomly but only from the elements in *Starts*.

Thus, our algorithm is:

```
1  Read inputs:
2  Calculate cost matrix, $C$ and precedence matrix, $P$.
3  Initialise:
   Set $t := 0$ {$t$ is the time counter}
   Set $NC := 0$ {$NC$ is the cycles counter}
   For $\tau_{ij}(t) = c$ and $\Delta\tau_{ij} = 0$ for every operation pair $(i, j)$
   Store the operations with no precedence constraints in Starts.
   Place $m$ ants on a starting operation randomly from elements Starts.
4  Set $s := 1$ {$s$ is the tabu list index}
   For $k := 1$ to $m$ do
   Place the starting operation of the kth ant in $tabu_k(s)$
5  Repeat until tabu list is full
   Set $s := s + 1$
   For $k := 1$ to $m$ do
   Choose operation $j$ after operation $i$ with probability $p_{ij}^k(t)$ {at time $t$ the kth ant is on
   operation $i = tabu_k(s - 1)$}
   Store the operations that must precede operation $j$ into Pre_j for all $j \in allowed_k$
   Check if all operations from Pre_j are present in $TABU_k$. If at least one operation is
   missing: set $p_{ij}^k(t) = 0$.
   Make the kth ant choose operation $j$ by roulette wheel selection.
   $j = tabu_k(s)$
6  For $k := 1$ to $m$ do
   Move the kth ant from $tabu_k(n)$ to $tabu_k(1)$
   Compute $C_k$
   Update the minimum cost sequence found.
   For every $(i, j)$th transition
   For $k := 1$ to $m$ do
```

```
     Calculate $\Delta\tau_{ijk}$; $\Delta\tau_{ij} = \Delta\tau_{ij} + \Delta\tau_{ij}^{k}$
 7   For every transition $(i, j)$ compute $\tau_{ij}(t + n)$
     Set $t := t + n$
     Set $NC := NC + 1$
     For every transition $(i, j)$ set $\Delta\tau_{ij} := 0$
 8   If ($NC < NC_{MAX}$) and ($ANB > ANB_{MIN}$) (not stagnation behaviour)
     then
     Empty all tabu lists
     Goto step 3
     else
     Print optimal sequence
     Stop.
```

The parameter values are set as: $NC_{MAX} = 5000$; $n = m$; $ANB_{\min} = 2$; $\alpha = 1$; $\beta = 5$; $\rho = 0.5$; $Q = 10$ (Dorigo et al., 1996).

5 Results and discussions

Two parts: part 1 and part 2 (see Figure 1 and Figure 5) are used as examples to illustrate the performance of our algorithm. The results obtained by application of our algorithm on optimisation of the operation sequences for the two parts are shown in the Sections 5.1 and 5.2, which is then followed by discussions in Section 5.3.

5.1 Results for part 1

Part 1 is a prismatic part that has 21 identifiable features which require a total of 28 machining operations to manufacture the part. Figure 1 shows part 1 for which the optimal sequence of machining operations have to be determined. Here, each operation requires one out of the six available tool approach directions. The six tool approach directions are defined as the directions normal to each of the six face planes (numbered 1 to 6 in Figure 2) and pointing outwards. Table 1 lists the features, machining operations required to produce the part, and the corresponding face plane numbers for part 1. The precedence constraints between machining operations for the part 1 are shown in Table 2.

Tables 1 and 2 constitute the input for our algorithm. Here any transition that requires a change in the operation type and/or the face plane number will require a change in the cutting tool and a change in the set-up respectively. The MATLAB programme for the algorithm, described in Section 4.2, ran till the 19th cycle at which a stagnation period was reached, i.e., the average node branching became less than 2 (see Figure 4), thus accounting for a total computation time of *11 seconds*. Figure 3 shows the history of the best cost, i.e., the cost for the best sequence among all the *m* sequences covered by the ants in a particular cycle. It also shows the history of the global best cost, i.e., the cost for the best sequence found from the start till the current cycle. An optimal sequence (global best) occurred multiple times with the first occurrence in the *fifth* ant cycle *within 3 seconds*.

Figure 1 Part 1: a prismatic part

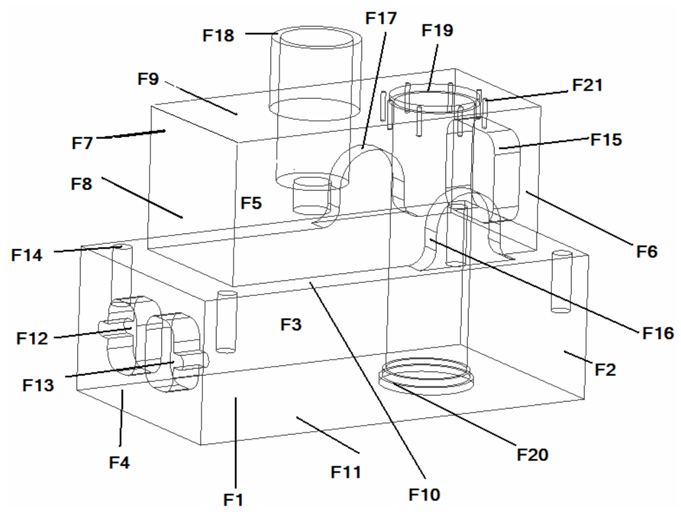


Figure 2 Convention followed in determining the face plane number and the corresponding tool approach direction for part 1 (see online version for colours)

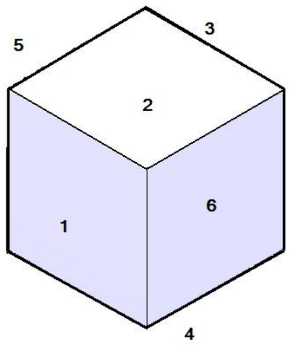


Table 1 The list of features, operations, and face plane numbers for part 1

Operation no.	*Operation type identifier*	*Feature no.*	*Face plane no.*	*Feature description*
1	Face milling	1	6	Planar surface
2	Face milling	2	3	Planar surface
3	Face milling	3	5	Planar surface
4	Face milling	4	1	Planar surface
5	Face milling	5	6	Planar surface
6	Face milling	6	3	Planar surface
7	Face milling	7	5	Planar surface
8	Face milling	8	1	Planar surface
9	Face milling	9	2	Planar surface
10	Face milling	10	2	Planar surface
11	Face milling	11	4	Planar surface

Table 1 The list of features, operations, and face plane numbers for part 1 (continued)

Operation no.	*Operation type identifier*	*Feature no.*	*Face plane no.*	*Feature description*
12	End milling	12	1	Rectangular pocket
13	End milling	13	1	Rectangular pocket
14	Drilling	14	2	Array of holes
15	Drilling	14	2	Array of holes
16	Drilling	14	2	Array of holes
17	Drilling	14	2	Array of holes
18	End milling	15	3	Rectangular pocket
19	End milling	16	6	Protrusion rib
20	End milling	17	5	Protrusion rib
21	End milling	18	2	Boss
22	Boring	18	2	Compound hole
23	Drilling	18	2	Compound hole
24	Boring	19	2	Compound hole
25	Drilling	20	4	Compound hole
26	Boring	20	4	Compound hole
27	Drilling	21	2	Array of holes
28	Tapping	21	2	Array of holes

Table 2 The list of precedence constraints between machining operations for part 1

Operation to be done before column 2	*Operation to be done after column 1*	*Description of the associated constraint*
11	All	Datum and support face of the part
25	26	Material removal interaction
2	6	Datum surface for the operation
6	18	Datum surface for the operation
4	12	Datum surface for the operation
4	13	Datum surface for the operation
4	8	Datum surface for the operation
3	7	Datum surface for the operation
20	7	Datum surface for the operation
1	5	Datum surface for the operation
19	5	Datum surface for the operation
9	21	Fixed order of machining
21	23	Fixed order of machining
23	22	Fixed order of machining
25	24	Fixed order of machining
27	28	Material removal interaction
10	9	Material removal interaction
10	14	Material removal interaction
10	15	Material removal interaction
10	16	Material removal interaction

Table 2 The list of precedence constraints between machining operations for part 1 (continued)

Operation to be done before column 2	*Operation to be done after column 1*	*Description of the associated constraint*
10	17	Material removal interaction
5	10	Material removal interaction
6	10	Material removal interaction
7	10	Material removal interaction
8	10	Material removal interaction
19	10	Material removal interaction
20	10	Material removal interaction
9	24	Datum surface for the operation
9	27	Datum surface for the operation
9	28	Datum surface for the operation
26	9	Fixturing interaction
26	10	Fixturing interaction
18	4	Fixturing interaction

Figure 3 History of best cost found at each cycle and global best cost at each cycle for part 1 (see online version for colours)

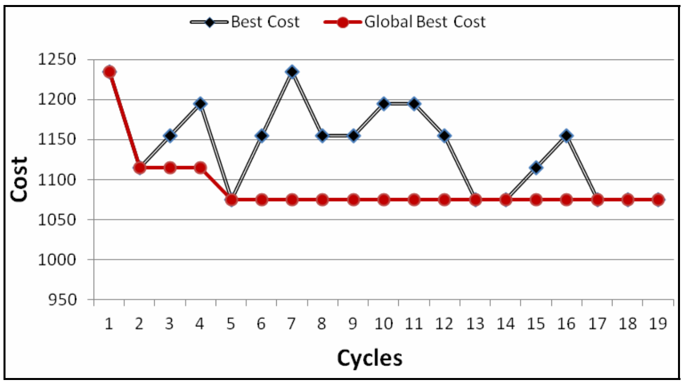


Figure 4 Average node branching at every cycle for part 1

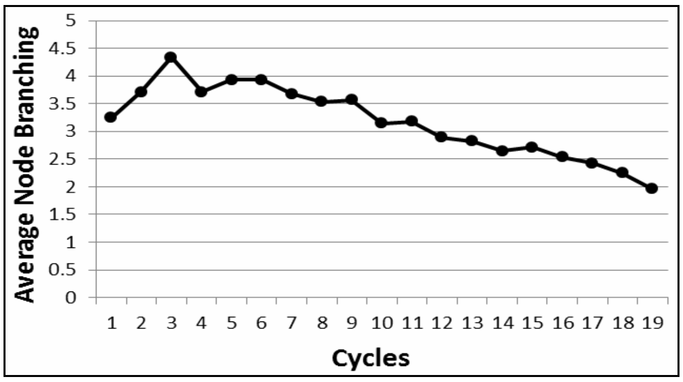


Table 3 shows a global best operation sequence found at the cycle number 5. From Table 3, we see that in the optimal sequence, the number of tool changes = 11 and the number of set-up changes = 5. The cost associated with it from equation (4) = 5 × 27 + 40 × 11 + 100 × 5 = *1,075 units*.

Table 3 Global best solution, i.e., the optimal sequence among all feasible operation sequences generated during run of the algorithm for part 1

Operation no.	*Operation type identifier*	*Feature no.*	*Face plane no.*	*Feature description*
11	Face milling	11	4	Planar surface
25	Drilling	20	4	Compound hole
26	Boring	20	4	Compound hole
2	Face milling	2	3	Planar surface
6	Face milling	6	3	Planar surface
18	End milling	15	3	Rectangular pocket
20	End milling	17	5	Protrusion rib
3	Face milling	3	5	Planar surface
7	Face milling	7	5	Planar surface
4	Face milling	4	1	Planar surface
8	Face milling	8	1	Planar surface
12	End milling	12	1	Rectangular pocket
13	End milling	13	1	Rectangular pocket
19	End milling	16	6	Protrusion rib
1	Face milling	1	6	Planar surface
5	Face milling	5	6	Planar surface
10	Face milling	10	2	Planar surface
9	Face milling	9	2	Planar surface
21	End milling	18	2	Boss
23	Drilling	18	2	Compound hole
27	Drilling	21	2	Array of holes
17	Drilling	14	2	Array of holes
16	Drilling	14	2	Array of holes
15	Drilling	14	2	Array of holes
14	Drilling	14	2	Array of holes
24	Boring	19	2	Compound hole
22	Boring	18	2	Compound hole
28	Tapping	21	2	Array of holes

5.2 *Results for part 2: a comparative study*

Part 2, shown in Figures 5 and 6 and Table 4, is taken from the case study 2 presented in Krishna and Rao (2006) as also in Irani et al. (1995). Before comparing the results we present a brief summary of the similarities and the differences between both approaches. Firstly, the algorithm used by us is based on ant system while the algorithm used by Krishna and Rao (2006) is based on the ant colony system. Thus, both the algorithms share the basic similarities and differences of ant system and ant colony system (refer to

Dorigo et al., 1996; Dorigo and Gambardella, 1997). In addition both the algorithms have different parameter settings ($\alpha = \rho = 0.1$; $\beta = 2$ for Krishna and Rao, 2006). However, a major structural difference lies in the method adopted to generate a feasible sequence. The algorithm proposed by Krishna and Rao (2006) assigns a very high cost to every operation transition that violates a precedence constraint, so that the probability of obtaining an infeasible sequence is reduced but is never zero. However, the algorithm proposed by us never generates any infeasible sequence as the precedence rules are checked at every iteration and the probability of an infeasible transition is made zero (refer to Section 4.2). This feature enables the algorithm to search for an optimal solution in much fewer iterations and much less computation time than the previous algorithm. To complement this, our algorithm chooses the starting node for every ant only from operations with no precedence restrictions. Krishna and Rao (2006) have considered relative costs for machine parameter change (similar to default cost in our case), tool change, set-up change and an additional machine change cost which are denoted by x1, x2, x3 and x4 in Table 4. Our algorithm can be adapted to the same with a mere addition of an extra term for machine change cost in the cost matrix calculation [equations (1) to (4)]. We have used the cost matrix for part 2 provided in Krishna and Rao (2006) to run our algorithm.

Figure 5 Part 2

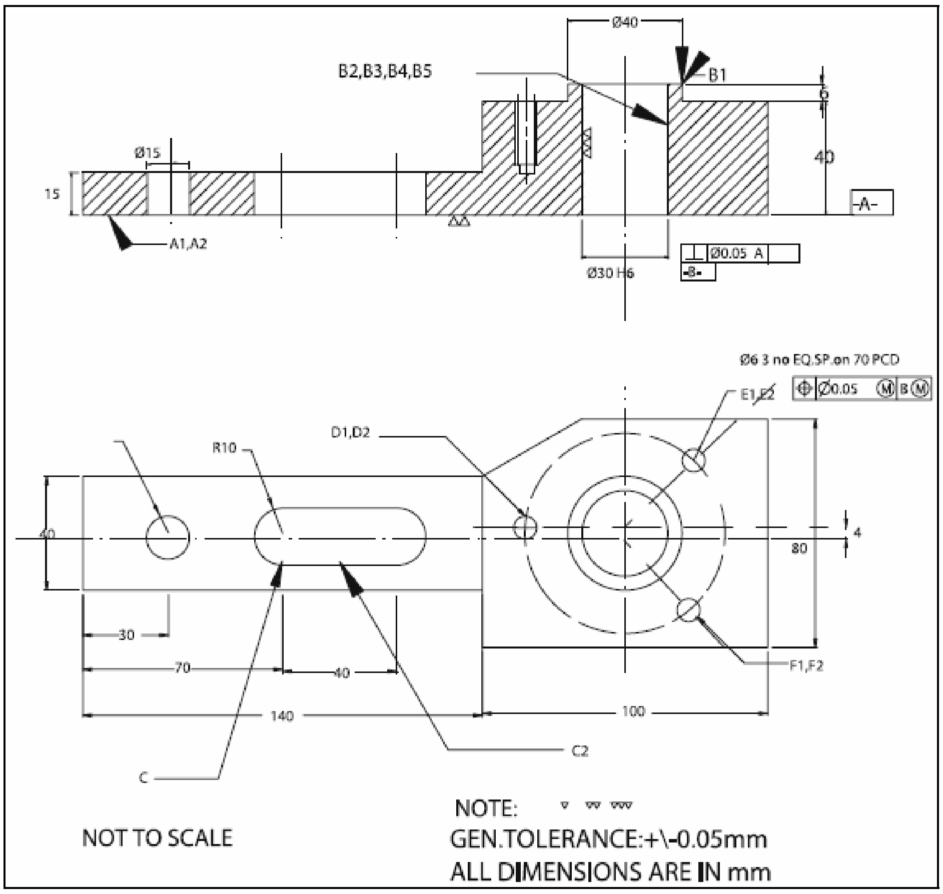
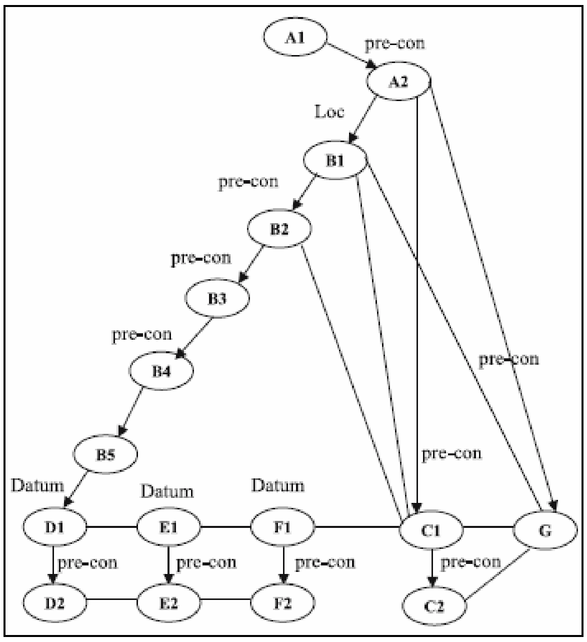


Figure 6 Operations precedence graph for the part 2: x1 = 1, x2 = 2, x3 = 10 and x4 = 999



Notes: A1 – rough plain milling; A2 – finish plain milling; B1 – milling the boss; B2 – drilling; B3 – boring; B4 – reaming; B5 – finish reaming; C1 – drilling; C2 – milling; D1, E1, F1, G – drilling; D2, E2, F2 – tapping.

Table 4 The cost matrix for part 2

	A1	*A2*	*B1*	*B2*	*B3*	*B4*	*B5*	*C1*	*C2*	*D1*	*D2*	*E1*	*E2*	*F1*	*F2*	*G*
A1	-	x1	x4	x4	x4	x4	x4	x4	x4	x4	x4	x4	x4	x4	x4	x4
A2	x4	-	x3	x4	x4	x4	x4	x4	x4	x4	x4	x4	x4	x4	x4	x4
B1	x4	x4	-	x2	x4	x4	x4	x2	x4	x4	x4	x4	x4	x4	x4	x2
B2	x4	x4	x4	-	x2	x4	x4	x2	x4	x4	x4	x4	x4	x4	x4	x2
B3	x4	x4	x4	x4	-	x2	x4	x4	x4	x4	x4	x4	x4	x4	x4	x4
B4	x4	x4	x4	x4	x4	-	x1	x4	x4	x4	x4	x4	x4	x4	x4	x4
B5	x4	x4	x4	x4	x4	x4	-	x2	x4	x2	x1	x2	x4	x4	x4	x2
C1	x4	x4	x2	x2	x4	x4	x4	-	x2	x2	x4	x4	x4	x4	x4	x2
C2	x4	x4	x4	x4	x4	x4	x4	x4	-	x2	x4	x4	x4	x4	x4	x2
D1	x4	x4	x4	x4	x4	x4	x4	x2	x4	-	x2	x2	x4	x4	x4	x2
D2	x4	x4	x4	x4	x4	x4	x4	x4	x4	x4	-	x2	x2	x2	x4	x4
E1	x4	x4	x4	x4	x4	x4	x4	x4	x4	x2	x4	-	x2	x2	x4	x4
E2	x4	x4	x4	x4	x4	x4	x4	x4	x4	x4	x2	x4	-	x2	x2	x2
F1	x4	x4	x4	x4	x4	x4	x4	x4	x4	x4	x2	x2	x2	-	x2	x2
F2	x4	x4	x4	x4	x4	x4	x4	x4	x4	x4	x4	x2	x2	x4	-	x2
G	x4	x4	x2	x2	x4	x4	x4	x2	x2	x2	x4	x4	x4	x2	x2	-

An important observation on running the algorithm was that during most of the runs a stagnation period was reached at the very first or the second cycle. This may be explained from the fact that part 2 has a small number of feasible solutions which is also indicated from the precedence graph in Figure 6. Using this average node branching; ANB, i.e., the average of the possible number of exit routes from each node/operation is calculated below. Here, NB_i is the node branching or the maximum possible exit routes from operation i.

$$\begin{aligned} ANB &= \left(\begin{array}{l} NB_{A1} + NB_{A2} + NB_{B1} + NB_{B2} + NB_{B3} + NB_{B4} + NB_{B5} + NB_{C1} \\ +NB_{C2} + NB_{D1} + NB_{D2} + NB_{E1} + NB_{E2} + NB_{F1} + NB_{F2} + NB_{G} \end{array} \right) \Big/ 16 \\ &= (1+3+3+2+1+1+1+1+5+1+2+1+3+2+3+1+3)/16 \\ &= 2.063. \end{aligned} \tag{10}$$

Thus, by the definition of average node branching for the ant system by Dorigo et al. (1996) the likelihood of minimum average node branching to reach below the limit of 2 is high, therefore to allow the algorithm to search for more sequences it was modified to check for a minimum number of cycle completions before termination. For demonstration we choose minimum number of five cycles before the programme starts checking for stagnation behaviour. With this modification, the MATLAB programme ran till the ninth cycle at which a stagnation was reached, i.e., the average node branching became less than 2 (see Figure 8), thus accounting for a total *computation time of only 2 seconds*. Figure 7 shows the comparison of the history of the 'best cost' and the 'global best cost' in every cycle. An optimal sequence (global best) occurred at the *third* cycle and is given as: $A1 \rightarrow A2 \rightarrow B1 \rightarrow C1 \rightarrow C2 \rightarrow G \rightarrow B2 \rightarrow B3 \rightarrow B4 \rightarrow B5 \rightarrow D1 \rightarrow D2 \rightarrow E1 \rightarrow F1 \rightarrow E2 \rightarrow F2$ having a cost of 36 units. On the other hand ACO-based algorithm by Krishna and Rao (2006) took 20 iterations and a computation time of 18 seconds to get an optimal sequence with same total cost of *36 units*. When GA is used with a population size of 40, crossover probability 0.9, and mutation probability 0.1; the execution time is 50 seconds (Krishna and Rao, 2006).

Figure 7 History of best cost found at each cycle and global best cost at each cycle for part 2 (see online version for colours)

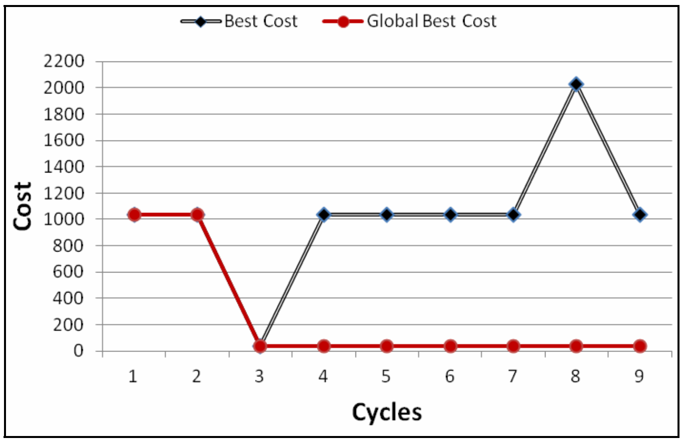
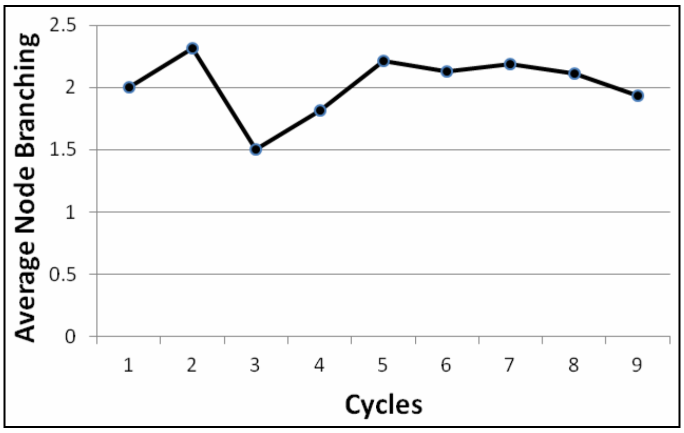


Figure 8 Average node branching at every cycle for part 2 (see online version for colours)



5.3 Discussions

It is observed that for simpler parts having lower values of node branching (less than or equal to 2) the search space of possible feasible sequences is less. In such cases on introducing a minimum number of cycles the algorithm proposed in the paper performs extremely fast. A comparison of results for part 2 shows that the programme performed *nine times faster* than previous methodology for ACO and *25 times faster* than a GA search technique. Even for a slightly more complex part 1 with 28 operations the performance remains very good and the total computation time for a demonstration run of the algorithm was 11 seconds, and the first global best solution occurred at fifth ant cycle.

Our interpretation of previous ant colony-based algorithm by Krishna and Rao (2006) is that the concept of assigning a high cost to an infeasible operation transition does not exclude any infeasible sequences from the solution search space and hence such an algorithm may generate high cost solutions, that are practically infeasible, at the beginning before converging towards the feasible ones, thereby unnecessarily increasing the computations. Since the algorithm proposed in this paper uses precedence checks at the transition rules therefore the solution search space is reduced as it now includes only feasible solutions. The logical validity of our approach is also evident from the fact that if we consider a part that is not feasible to make considering the precedence constraints, then the algorithm developed by previous researchers would still generate a sequence which would be infeasible. However, our proposed method would lead to the generation of an error message indicating that such a part can never be manufactured. Therefore, from the result and discussions, it can be said with considerable confidence that the proposed approach is an improvement over the previously existing techniques.

In addition the method is flexible and can be used for any type of manufacturing part like, rotational, prismatic, etc. In addition, the optimisation criteria can be modified easily by adding parameters like machine change, tool travel distance data or removing parameters like tool change or set-up change data, in order to suit different manufacturing conditions in a CAPP system. Cost calculation can also be modified according to the chosen optimisation criteria and using the actual cost factors for a particular plant. Similarly, list of constraints can be extended to include clustering rules, adjacent order rules and so on, by adding the corresponding input data and augmenting more checks at every transition.

6 Conclusions

This paper demonstrates an algorithm that uses ant system as a metaheuristic for solving the problem of automatically finding the optimum machining operation sequence in CAPP to minimise the total manufacturing cost. The comparison of the algorithm with the existing evolutionary computing techniques shows that proposed algorithm performs nearly nine times faster and 25 times faster respectively than previous methodologies based on ant colony and genetic algorithm. This is probably due to the unique features of the algorithm, i.e., the concept of *specific selection of a starting node at the beginning of each ant cycle and introducing a precedence check in the transition rules*, which prevent ant from making infeasible sequences. The specific selection of the starting node restricts each ant to start a sequence directly from the machining operation with no precedence constraint (like datum surface of a part). The precedence check sets the probability of any transition that will generate infeasible sequence as zero.

Although the paper illustrates application of algorithm to optimisation of operation sequences based on minimising the number of tool changes and set-up changes, the algorithm can also use other optimisation criteria, like minimisation of machine changes and tool travel distances, or optimisation of multiple objectives. Furthermore, depending on the problem at hand, the constraints can be extended to include clustering rules, adjacent order rules, and other machining rules in addition to the precedence rules. Considering the significantly reduced computation times and flexibility of its application, the proposed adaptation of ant system is clearly an improvement over the previous approaches to operation sequencing.

Being developed based on the ant system, the algorithm suffers from the same basic limitation as the other ant colony algorithms, i.e., it employs random search thus, although the convergence of solution space is guaranteed, the convergence rate may decrease with increase in the problem complexity. Therefore, it is important to test this algorithm on higher order problems say, with more than 50 operations. Future research is also required to verify that modifications in the transition rules to accommodate constraints, like those implemented in this paper, may lead to significant improvement in computations of ant colony algorithms.

Acknowledgements

The authors would like to sincerely thank the reviewers for their constructive and helpful recommendations to revise this paper.

References

Azadeh, A., Keramati, A., Karimi, A. and Moghaddam, M. (2010) 'A multi-objective genetic algorithm for scheduling optimisation of m job families on a single machine', *Int. J. Industrial and Systems Engineering*, Vol. 6, No. 4, pp.417–440.

Azizi, N., Liang, M. and Zolfaghari, S. (2009) 'Hybrid simulated annealing in flow-shop scheduling: a diversification and intensification approach', *Int. J. Industrial and Systems Engineering*, Vol. 4, No. 3, pp.326–348.

Blum, C. and Dorigo, M. (2004) 'The hyper-cube framework for ant colony optimization', *IEEE Transactions on Systems, Man, and Cybernetics: Part B*, Vol. 34, No. 2, pp.1161–1172.

Bo, Z.W., Hua, L.Z. and Yu, Z.G. (2006) 'Optimization of process route by genetic algorithms', *Robotics and Computer-Integrated Manufacturing*, Vol. 22, No. 2, pp.180–188.

Bullnheimer, B., Hartl, R.F. and Strauss, C. (1999) 'A new rank based version of the ant system: a computational study', *Central European Journal for Operations Research and Economics*, Vol. 7, No. 1, pp.25–38.

Dereli, T. and Filiz, I.H. (1999) 'Optimisation of process planning functions by genetic algorithms', *Computers & Industrial Engineering*, Vol. 36, No. 2, pp.281–308.

Dharmapriya, U.S.S., Siyambalapitiya, S.B. and Kulatunga, A.K. (2012) 'A new hybrid algorithm for multi-depot vehicle routing problem with time windows and split delivery', *Int. J. Industrial and Systems Engineering*, Vol. 11, Nos. 1–2, pp.110–121.

Dorigo, M. and Gambardella, L.M. (1997) 'Ant colony system: a cooperative learning approach to the traveling salesman problem', *IEEE Transactions on Evolutionary Computation*, Vol. 1, No. 1, pp.53–66.

Dorigo, M., Maniezzo, V. and Colorni, A. (1996) 'The ant system: optimization by a colony of cooperating agents', *IEEE Transactions on Systems, Man, and Cybernetics: Part B*, Vol. 26, No. 1, pp.1–13.

Gagne, C., Price, W.L. and Gravel, M. (2002) 'Comparing an ACO algorithm with other heuristics for the single machine scheduling problem with sequence-dependent setup times', *Journal of the Operational Research Society*, Vol. 53, No. 8, pp.895–906.

Goldberg, D.E. (1989) *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, New York.

Guo, Y.W., Mileham, A.R., Owen, G.W. and Li, W.D. (2006) 'Operation sequencing optimization using a particle swarm optimization approach', *Proceedings of the Institution of Mechanical Engineers: Part B*, Vol. 220, No. 12, pp.1945–1958.

Hua, G.R., Zhou, X.H. and Ruan, X.Y. (2007) 'GA-based synthesis approach for machining scheme selection and operation sequencing optimization for prismatic parts', *International Journal of Advanced Manufacturing Technology*, Vol. 33, Nos. 5–6, pp.594–603.

Irani, S.A., Koo, H.Y. and Raman, S. (1995) 'Feature based operation sequence generation in CAPP', *International Journal of Production Research*, Vol. 33, No. 1, pp.17–39.

Iredi, S., Merkle, D. and Middendorf, M. (2001) 'Bi-criterion optimization with multi colony ant algorithms', *Lecture Notes in Computer Science*, Vol. 1993, pp.359–372.

Krishna, A.G. and Rao, K.M. (2006) 'Optimization of operations sequence in CAPP using an ant colony algorithm', *International Journal of Advanced Manufacturing Technology*, Vol. 29, Nos. 1–2, pp.159–164.

Kumar, C. and Deb, S. (2012) 'Generation of optimal sequence of machining operations in set-up planning by genetic algorithms', *Journal of Advanced Manufacturing Systems*, Vol. 11, No. 1, pp.67–80.

Kundu, A. and Dan, P.K. (2012) 'Metaheuristic in facility layout problems: current trend and future direction', *Int. J. Industrial and Systems Engineering*, Vol. 10, No. 2, pp.238–253.

Li, L., Fuh, J.Y.H., Zhang, Y.F. and Nee, A.Y.C. (2005) 'Application of genetic algorithm to computer-aided process planning in distributed manufacturing environments', *Robotics and Computer-Integrated Manufacturing*, Vol. 21, No. 6, pp.568–578.

Li, W.D., Ong, S.K. and Nee, A.Y.C. (2002) 'Hybrid genetic algorithm and simulated annealing approach for the optimization of process plans for prismatic parts', *International Journal of Production Research*, Vol. 40, No. 8, pp.1899–1922.

Ma, G.H., Zhang, Y.F. and Nee, A.Y.C. (2000) 'A simulated annealing-based optimization algorithm for process planning', *International Journal of Production Research*, Vol. 38, No. 12, pp.2671–2687.

Pawar, P.J., Rao, R.V. and Davim, J.P. (2010) 'Multiobjective optimization of grinding process parameters using particle swarm optimization algorithm', *Materials and Manufacturing Processes*, Vol. 25, No. 6, pp.424–431.

Qiao, L., Wang, X.Y. and Wang, S.C. (2000) 'A GA-based approach to machining operation sequencing for prismatic parts', *International Journal of Production Research*, Vol. 38, No. 14, pp.3283–3303.

Rahim, A. and Shakil, M. (2011) 'A tabu search algorithm for determining the economic design parameters of an integrated production planning, quality control and preventive maintenance policy', *Int. J. of Industrial and Systems Engineering*, Vol. 7, No. 4, pp.477–497.

Rameshkumar, K., Rajendran, C. and Mohanasundaram, K.M. (2011) 'Discrete particle swarm optimisation algorithms for minimising the completion-time variance of jobs in flowshops', *Int. J. Industrial and Systems Engineering*, Vol. 7, No. 3, pp.317–340.

Rao, R.V., Pawar, P.J. and Davim, J.P. (2010) 'Parameter optimization of ultrasonic machining process using nontraditional optimization algorithms', *Materials and Manufacturing Processes*, Vol. 25, No. 10, pp.1120–1130.

Reddy, S.V.B., Shunmugam, M.S. and Narendran, T.T. (1999) 'Operation sequencing in CAPP using genetic algorithms', *International Journal of Production Research*, Vol. 37, No. 5, pp.1063–1074.

Salehi, M. and Bahreininejad, A. (2011) 'Optimization process planning using hybrid genetic algorithm and intelligent search for job shop machining', *J. Intell. Manuf.*, Vol. 22, No. 4, pp.643–652.

Salehi, M. and Tavakkoli-Moghaddam, R. (2009) 'Application of genetic algorithm to computer-aided process planning in preliminary and detailed planning', *Engineering Applications of Artificial Intelligence*, Vol. 22, No. 8, pp.1179–1187.

Singh, G. and Singh, A. (2012) 'Review paper: operation sequencing using genetic algorithm', *International Journal of Emerging trends in Engineering and Development*, Vol. 5, No. 2, pp.863–867.

Stützle, T. and Hoos, H.H. (2000) 'MAX-MIN ant system', *Future Generation Computer Systems*, Vol. 16, No. 8, pp.889–914.

Xu, X., Wang, L. and Newman, S.T. (2011) 'Computer-aided process planning – a critical review of recent developments and future trends', *International Journal of Computer Integrated Manufacturing*, Vol. 24, No. 1, pp.1–31.