Mechanical Engineering Research; Vol. 4, No. 2; 2014
ISSN 1927-0607 E-ISSN 1927-0615
Published by Canadian Center of Science and Education

CNC Machining Path Planning Optimization for Circular Hole Patterns via a Hybrid Ant Colony Optimization Approach

Adel T. Abbas[1], Karim Hamza[2] & Mohamed F. Aly[3]

[1] Department of Mechanical Engineering, King Saud University, Riyadh, Saudi Arabia

[2] Department of Mechanical Engineering, University of Michigan, Ann Arbor, USA

[3] Department of Mechanical Design, American University in Cairo, New Cairo, Egypt

Correspondence: Adel T. Abbas, Department of Mechanical Engineering, King Saud University, Saudi Arabia, 11421, P.O.Box 800. Tel: 966-1467-9799. E-mail: aabbas@ksu.edu.sa

Received: August 14, 2014 Accepted: September 2, 2014 Online Published: September 5, 2014

doi:10.5539/mer.v4n2p16 URL: http://dx.doi.org/10.5539/mer.v4n2p16

Abstract

This paper presents a path-planning optimization study for a Computer Numerical Control (CNC) machining center tasked with machining jobs involving a large number of holes to drill that are mostly arranged in concentric circular patterns. Benefits of this research may contribute to shortening the machining time in certain components used in heat exchangers, boilers, condensers, trammel screens and food separators. Optimization of tool travel distance and machining cost are typically overlooked aspects when generating tool paths and CNC codes from commercially available CAD software packages. Tool path travel distance minimization can be modelled Travelling Salesman Problem (TSP). Optimization algorithms have been heavily applied in the literature to the TSP with varying levels of success. Ant Colony Optimization (ACO) is one of the most prominent approaches that mimics the natural behavior of ant colonies. The research in this paper proposes a hybrid ACO that has a biasing mechanic designed to take advantage of the geometric hole-pattern arrangement, as well as a local search. Simulation examples show the proposed approach exhibiting superior performance compared to the classic ACO approach, a genetic algorithm (GA) approach, as well as the simple spiral path generated via commercial CAD software. The proposed approach is then applied to the drilling path planning of a two-thousand-hole food-industry separator plate.

Keywords: CNC path planning, automated drilling operations, ant colony optimization, genetic algorithm

1. Introduction

Total machining time and cost of production in Computer Numerical Control (CNC) drilling operations of large number of holes arrangements is of a significant concern to manufacturing companies. Associated time investment in process optimization is often well justified. The fraction of machining time spent on drilling holes can vary from an average of 40% for a typical product (Shunmugam et al., 2000) to more than 80% for products that incorporate dense matrices of holes such as heat exchangers, food-processing separators (Figure 1), as well as drum and trammel screens. And as such, cost optimization of CNC drilling operations has been an important field of study in the literature (Ke et al., 2006; Zhu & Zhang, 2008; Satishkumar & Asokan, 2008). Along with the opportunity of reducing the production via small improvements in the cutting conditions, the total travelled distance of the tool between drilling locations can be reduced through optimum path planning. Such reduction can account for up to 20% of the total machining time of the part, which in turn can have a significant impact on cost.

Path planning to minimize travel time between drilling locations can be formulated as a Travelling Salesman Problem (TSP), which is classified as NP-hard, well known and difficult to solve in integer programming (Wolsey & Nemhauser, 199). Several approaches have been applied to TSP including: Dynamic Programming (Denardo, 2004), Genetic Algorithms (Goldberg, 1989; Michalewiz, 1992; Anand et al., 1999), Simulated Annealing (Aarts, 1989; Van Laarhoven & Aarts, 2009), Particle Swarm Optimization (Zhu, 2006), and Ant Colony Optimization (ACO) (Dorigo, 1996; Stutzle & Dorigo, 2002; Dorigo & Stutzle, 2004; Dorigo et al., 2004; Zhou, 2009; Arnaout et al., 2010), whose run-time modeling (Stutzle & Dorigo, 2002; Dorigo et al., 2004) suggests it as one of the most efficient algorithms to address TSP types of problems. After settling on optimum paths, it is a fairly straight-forward task to automatically generate the G-code for CNC machining centers (Abbas & Megahed, 2005;

Abbas, 2012). Applications of modeling drilling operations as TSP include food-industry separators (Abbas et al., 2011) and printed circuit boards (Montiel-Ross et al., 2012; Saealal et al., 2012).

Figure 1. Food-industry processing separators

This paper extends the approach in (Abbas et al., 2011) with a focus on the optimization of a CNC drilling tool path between holes lying in concentric circular patterns. Simple examples studies as well as a full-scale industrial application of a food processing separator screen are tested. Results of the studies show better performance of the modified ACO algorithm compared to the classic ACO, the modified version in (Abbas et al., 2011), a typical genetic algorithm, as well as establish the scalability to large problems.

The paper starts with a brief introduction and a review of relevant work reported in the literature that highlights the motivation to carry out this research. The formulation of the optimization problem with details of the classic ACO algorithm and the proposed modifications is then presented. Example studies and an industrial application are then presented to demonstrate the algorithms' performance. The paper then concludes with a brief discussion and prospective future extensions.

2. Problem Formulation

Tool path planning for the CNC drilling tool is modelled as a TSP when ignoring the tool dynamics (which is a reasonable assumption when the tool has to dwell a reasonable amount of time in each visited location to be drilled). TSP can be formulated as an integer linear program (Padberg & Sung, 1991) as follows:

$$\text{Minimize:} \quad f(x) = \sum_{i}^{n} \sum_{j}^{n} X_{ij} D_{ij} \tag{1}$$

$$\text{Subject to:} \quad \sum_{i=1}^{n} X_{ij} = 1, j = 1, \ldots, n \tag{2}$$

$$\sum_{j=1}^{n} X_{ij} = 1, i = 1, \ldots, n \tag{3}$$

$$\sum_{i=k}^{k+l-1} \sum_{j=k}^{k+l-1} X_{ij} < l, k = 1, \ldots, n\text{-}l\text{+}1, l = 1, \ldots, n\text{-}1 \tag{4}$$

$$n = n_h + 1 \tag{5}$$

where:

n_h is the total number of holes

n is the total number of visited location by the drilling tool, which is equal to the number of holes to be drilled, plus the home position of the tool

D_{ij} is the travel distance from location i to location j

$X_{ij} \in \{0, 1\}$ are the design variables used to describe the tool path. $X_{ij} = 1$ means that the tool will travel from location i to location j as part of the path that loops on all the holes in the matrix. Otherwise, $X_{ij} = 0$ means that the path from location i to location j is not a part of the tool path.

The objective function is formulated as the summation of all the distances, between holes, travelled by the tool in the chosen path (Equation (1)). A minimization procedure is ran on this objective until an optimized path is selected. The set of constraints described in Eqn. (2), ensure that each hole j is only visited once in the path described by Xij, while the set of constraints described in Equation (3), ensure that the path coming out of every hole i goes to one other hole j. Further discussion of the integer linear programming formulation of TSP may be found in (Padberg & Sung, 1991) .Elements of the distance matrix Dij are calculated as the in-plane distance between the node centers as:

$$D_{ij} = D_{ji} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \tag{6}$$

where:

x_i is the coordinate position of location i along the X-axis

y_i is the coordinate position of location i along the y-axis

x_j is the coordinate position of location j along the X-axis

y_j is the coordinate position of location j along the y-axis

3. Path Planning Algorithms

CNC drilling tool path planning in this paper is formulated as a TSP in section 2 of this paper. Of the popular algorithms for solving TSP, Ant colony optimization was reported in the literature (Dorigo, 1996; Stutzle & Dorigo, 2002; Dorigo & Stutzle, 2004; Dorigo et al., 2004) as exhibiting superior performance in discovering globally optimal (or close to globally optimal) solutions within reasonable computational time. The work in this paper also introduces a modification to the basic ACO in order to take advantage of the geometrical arrangement of the holes to be drilled. Another popular algorithm used in several continuous and discrete optimization problems (TSP included) is genetic algorithm (GA) Goldberg (1989); Michalewiz (1992); Anand et al. (1999). This section explains the details of the algorithms tested in this paper.

3.1 Basic Ant Colony Optimization Algorithm

The classic Ant-Colony Optimization (ACO) (Dorigo et al., 2004) technique is naturally inspired from ants driven strategy to always find the shortest route from among multiple possible paths between their nest and food location. The basic mechanics behind the achievement of such path is realized through the following steps:

- Different ants start by trying different paths.
- As the ants travel through different paths they deposit a chemical compound known as *pheromone* along each path.
- Subsequent ants traveling along each of those paths have the ability to smell the pheromone and to make decisions on which path to take
- Ants choosing between two paths are *more likely* to follow the one with more pheromone deposit (with some randomness to this decision)
- Deposited pheromone tends to evaporate over time. Eventually the best path will be the most travelled one, and thus richest in pheromone

It should be noted that ACO (and GA, which is later considered in this paper) formulate the design variables $\boldsymbol{X}$ as permutation of a sequence of numbers between 1 and n (corresponding to the order of visiting the path nodes, which implicitly satisfies the constraints in Eqns. 2-4), rather than integer formulation. Main steps of a basic ACO algorithm (Dorigo et al., 2004) are described as:

```
Ant-Colony-TSP(n, D_ij)
1   P ← (1/n) × ones(n, n)          ▷ Initial pheromone distribution is uniform
        ▷▷ (equally likely for an ant at any location to choose          ▷▷ to go to
any other location)
2   X* ← Generate-Path(n, P)        ▷ An initial solution is recorded
3   while Termination-Condition() = FALSE
4       do  i ← 1
5       while i < nAnts
6           do  AntPaths ← φ
7               AntPaths ← AntPaths ∪ Generate-Path(n, P)
8               i ← i + 1
9           X^o ← Min-Distance(AntPaths)
10          if f(X^o) < f(X*)
11              then X* ← X^o
12          P ← Update-Pheromone(AntPaths, P)
13  return      X*
```

where:

P is the $n \times n$ *Pheromone probability matrix*, for path choices to be made by ants. P_{ij} is an indicator of the probability that an ant currently located at i will choose to go to location j next

`Termination-Condition()` is chosen as a fixed number of iterations, after which, the algorithm returns its best discovered path. Alternative termination criteria are used by other approaches such as the examination of the pheromone matrix P to invoke termination of the search when only one path becomes predominant in its pheromone content. This can only be applicable for problems with no local optima as the termination condition may never be met.

`Generate-Path()` is a sub-function in the ACO algorithm that generates a randomized path that visits all the target locations, with probability to select a path according to the distribution of the pheromone matrix P. Without loss of generality in the interpretation of the TSP path, the path always starts at the node corresponding to the tool home point. The probability of going from node i to node j is equal to:

$$\Pr(i \to j) = \frac{Q_j}{\sum_{j=1}^{n} Q_j} \tag{7}$$

$$\text{where } Q_j = \begin{cases} 0 & \text{if } j \text{ has already been visited} \\ P_{ij} & \text{otherwise} \end{cases} \tag{8}$$

`Update-Pheromone()` is a sub-function in the ACO algorithm that updates the pheromone matrix P. This incorporates that only ants that discovered the few best- paths will be the ones allowed to deposit pheromone on these paths, hence making it more likely for other ants to follow similar path choices.

The main steps of the function `Generate-Path()` for generating randomized paths are described as:

```
Generate-Path(n, P)
1   X ← φ
2   X ← X ∪ {1}            ▷ Always start at first node
3   while |X| < n
4       do  i ← X_|X|
5           for  j = 1 to n
6               do  Q_j ← P_ij
7           for  j = 1 to |X|
8               do  Q_Xj ← 0
9           S ← Sum(Q_j, j = 1 … n)
10          for  j = 1 to n
11              do  Q_j ← Q_j / S
12          nextNode ← Select(n, Q)   ▷ Selects a random integer between 1 to n (uniform distribution)
                                      ▷▷ with probability of selection Q_j
13          X ← X ∪ {nextNode}
14  return  X
```

The function `Update-Pheromone()` updates the pheromone matrix by calling a `Simple-Update-Pheromone()` sub-function several times but with different values of pheromone deposition of the few current best known paths. The `Simple-Update-Pheromone()` sub-function is used to update the current pheromone matrix via a path $\boldsymbol{X}$ and a deposited pheromone value of σ as follows:

```
Simple-Update-Pheromone(X, P, σ, nBestPaths)
1   k ← 1
2   i ← X_k
3   k ← k + 1
4   while    k < nBestPaths        ▷ depositing pheromone
5       do   j ← X_k
6            P_ij ← P_ij + σ
7            i ← j
8            k ← k + 1
9   for  i = 1 to n                ▷ normalizing P and diminishing the older pheromone values
10      do   S ← Sum(P_ij, j = 1 … n)
11           for  j = 1 to n
12               do  P_ij ← P_ij / S
13  return   P
```

It should be noted that the basic ACO (Dorigo et al., 2004) includes an explicit step for evaporation of existing pheromone before new pheromone is deposited. In this implementation however, the pheromone evaporation happens implicitly via the re-normalization of the pheromone matrix after the new pheromone is deposited.

3.2 Modified Ant Colony Optimization Algorithm

The basic ACO algorithm described in section 3.1 uses a uniform distribution for the initialization of the pheromone matrix P. At the start of the algorithm, an ant at some location i is equally likely to choose to visit any of the other locations j. Taking into consideration the special holes layout (holes with certain geometrical patterns such as concentric circles), it may not make much sense for an ant to travel from one end of a circle to its diametric opposite. As such, a modified initialization of the pheromone matrix (line #1 in the basic ACO algorithm discussed in section 3.1) follows along approaches that use heuristics to bias the probability of the choices made by the ants (Dorigo & Stutzle, 2004). The modification seeks to reduce the probability of selecting a step to the next hole if there are other closer choices:

$$P_{ij}^{N} = \begin{cases} \dfrac{1}{D_{ij}^{p}} & \text{if } D_{ij} \neq 0 \\ 0 & \text{if } D_{ij} = 0 \end{cases} \tag{9}$$

$$P_{ij} = \frac{P_{ij}^{N}}{\sum_{j=1}^{n} P_{ij}^{N}} \tag{10}$$

where:

p is an exponent for tuning the effect of the distance between the locations to be visited on the probability that ants make the choice to travel between them

P_{ij}^{N} is an un-normalized pheromone value that is designed to have higher values when the locations to be travelled between are close to one another. Normalization of P_{ij}^{N} via Eqn. 9 gives the pheromone value to be used in the ACO algorithm

It is noted that the proposed modification is generalization of both the basic ACO and the approach in (Abbas et al., 2011). Setting the exponent p in Eqn. 8 to zero causes the initial pheromone matrix to have equal values for travel between any locations (basic ACO), and setting p to a *very large number* (ACO would only consider immediate neighbors) equates to the approach in (Abbas et al., 2011). In some perspective, instead of using a heuristic matrix to influence the decision making of the ants as in (Dorigo & Stutzle, 2004), the perception that near neighbors are likely better choices is directly implemented in the initial pheromone matrix. As such, the proposed approach is perceived to have the following advantages:

a) No a-priori knowledge of a good path is required

b) Better computational efficiency (fewer terms to compute in the calculation of the probability of choosing a node)

c) Heuristic perception of good pathing automatically fades in favor of learned pathing as the search progresses (with pheromone updating) without the new for an additional tunable control parameter

3.3 Genetic Algorithm

The genetic algorithm tested in this paper is based on one of the implementations specific to TSP described in (Michalewiz, 1992, Chapter 10). Basic steps of a GA (Goldberg, 1989) are:

1. Initialize a random "Population" of solutions
2. Best known solution is recorded separately
3. Generate new solutions from existing population of solutions via: i) Selection, ii) Crossover, and iii) Mutation, and then place the newly generated solutions in a new population
4. When the number of solutions in the new population reaches the same size as current population, the new population replaces the current one, and the algorithm repeats at step 2
5. Steps 2-4 are repeated for a prescribed number of generations, then the best encountered solution is returned

Different implementations of GA will assure the maintenance of the overall algorithm structure, but will differ in the way a solution is represented, or the way new solutions are generated via selection, crossover and mutation. The setting for the GA tested in this paper is as follows:

3.3.1 Solution Representation: Path Representation

As in (Michalewiz, 1992, Chapter 10), the chromosome is encoded as an ordered string of integers that represent the sequence by which the nodes are visited in the TSP. For example, a parent chromosome P1: {1 2 3 4 5 6 7 8 9} means the start is at node 1, then node 2 is visited next, then node 3… until the last node, then a return from node 9 to node 1.

3.3.2 Selection: Binary Tournament

Selection of solutions from existing population to generate new solutions via crossover and mutation depends on the "fitness" of current solutions. In binary tournament selection (Michalewiz, 1992), two solutions are randomly picked from the current population, then the one that has better fitness (less travel distance in TSP) "wins the

tournament" and becomes a candidate parent. Another tournament is conducted to select a second parent, and then two child solutions are generated via crossover and mutation from the parent chromosomes.

3.3.3 Crossover: Order of Appearance

Crossover is the main operator that GA uses to generate new solutions from existing ones. The Order of Appearance Crossover (Michalewiz, 1992, Chapter 10) uses a randomly located (with uniform distribution) sub-segment of the chromosome of one parent, and then fills the rest of the newly generated chromosome via nodes chosen from another parent according to their order of appearance.

Two parent chromosomes P1 and P2 with segments to be exchanged are demonstrated as:

P1: {1 2 3 | 4 5 6 7 | 8 9}

P2: {4 5 2 | 1 8 7 6 | 9 3}

First, the segment to be exchanged is taken from first parent chromosome:

C1: {x x x | 4 5 6 7 | x x}

Then, the empty location in the new child chromosome is filled with the nodes that do not exist yet on the chromosome (i.e. nodes: 1, 2, 3, 8, 9) according to their order of appearance in the second chromosome:

C1: {2 1 8 | 4 5 6 7 | 9 3}

3.3.4 Mutation: Reciprocal Exchange

Newly generated child chromosomes have small probabilities of a mutation occurring. In reciprocal exchange (Michalewiz, 1992, Chapter 10), mutation randomly selects (with uniform distribution) two nodes in the chromosome and exchanges their spots.

3.4 Hybridization with Local Search

A known drawback of basic ACO and GA is that the later stages of convergence to an optimum (after a quasi-optimum solution is discovered) may take a long time. One approach discussed in the literature is to conduct local search within the global optimization (Bonabeau et al., 1999), however, in earlier experimentation with study problems, premature convergence of the global search was a concern. Alternatively, another common approach was adopted, which is to perform a local optimization search after ACO or GA is done, while using the solution returned by ACO/GA as a starting point for the location search.. A popular local search for TSP is based on Greedy search. The version adopted in (Abbas et al., 2011) is explained as:

```
Local-Search-TSP(n, X^G, D_ij)
1   X* ← X^G                ▷ Initialize best solution with the optimized solution
    ▷▷ returned from ACO or GA
2   for i = 1 to n
3       for j = i + 1 to n
4           X^t ← Swap-Order(X*, i, j)
5           if f(X^t) < f(X*)
6               then X* ← X^t
7   return      X*
```

Where $\boldsymbol{X}^{G}$, $\boldsymbol{X}^{*}$, $\boldsymbol{X}^{t}$ are respectively the initial path, final path and a temporary test path. The function `Swap-Order()` in line 4 simply swaps the order of two node locations i and j on the travel path. If the swap is beneficial (less overall travel distance), the swap is retained, otherwise, the swap is discarded.

4. Example Studies

4.1 Problem # 1: Three Complete Circular Patterns

A simple case study of a three concentric circular patterns (Figure 2) is investigated to evaluate the performance of the proposed algorithm. In the upcoming studies, tool velocity transients are not considered, thus the shortest distance tool path is also the one having the minim machining time. In all tests, a local search (as described in section 3.4) is conducted after the completion of each run of the tested algorithm. Comparison is conducted between 10 runs of each of: i) basic ACO ($p = 0$), ii) modified ACO with various settings of the exponent p, and iii) genetic algorithm. The results from various algorithms are also compared verses the typical spiral path (Figure 3) that is generated by most CAD software.

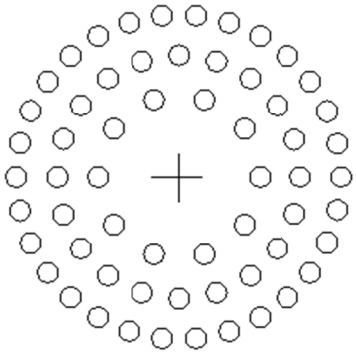

Figure 2. Circular pattern with three concentric circles

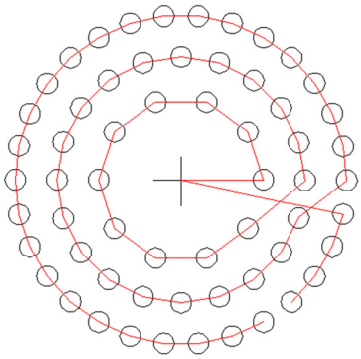

Figure 3. Typical spiral path generated by most CAD software

4.2 Problem # 2: Four Incomplete Circular Patterns

Similar numerical experiments for comparison of the tested algorithms discussed in section 4.1 are conducted for a machining task with more complex geometry (Figure 4), which could represent a more complex design of a food separator screen; one that will have handles at top and bottom portions of the plate and a pipe passing through the right side section.

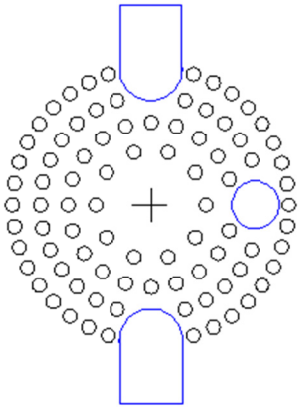

Figure 4. Circular pattern with four concentric circles and three zones vacated for design considerations

4.3 Results and Discussion

Details of the holes geometrical layout for test problems 1 and 2 are listed in Table 1. Tuning parameters for the ACO and GA (based on general recommendations in (Michalewiz, 1992; Dorigo et al., 2004)) are listed in Tables 2 and 3 respectively. While in general, an optimization problem with larger search space requires a larger GA population, it was found in earlier experimentation with problem #2 that using a population size of 100 with 60 generations was producing better results for GA than 120×50, so 100×60 was the chosen setting in order to present the best obtained results from GA.

Table 1. Geometrical layout of the example study problems

Problem #	1	2
Number of Circles	3	4
Inner Pitch Circle Diameter (mm)	40	40
Pitch Circle Diametric Increment (mm)	20	20
Total number of holes	60	86
Length of Spiral Path (mm)	625	944

Table 2. Tuning Parameters for ACO

Problem #	1	2
Number of Ants (*nAnts*)	50	60
Number of iterations until termination	100	100
Number of ants that update the pheromone matrix (*nBestPaths*)	5	5
Pheromone value used in updating the pheromone matrix (σ)	0.1	0.1

Results of 10 runs of the tested algorithms are plotted in Figure 5 and Figure 6 for problems #1 and #2 respectively. Numerical values of the results are also listed in Tables 4 and 5. Travel paths for one of the best obtained solutions for each problem are plotted in Figure 7. A general observation was GA did not exhibit very good performance and was not able to discover better paths than the simple spiral path. This may be due to the size of the TSP problems considered in the example studies; problem #1 has 61 nodes and problem #2 has 87 nodes. Such relatively large number of nodes often makes it difficult for GA to discover the optimum path, even when local search is invoked after the GA runs.

Table 3. Tuning Parameters for GA

Problem #	1	2
Population size	100	100
Number of Generations until termination	50	60
Crossover Probability	0.8	0.8
Mutation Probability	0.1	0.1

Table 4. Results summary for Problem #1

Algorithm	Average Path Length (mm)	Min.	Max.	Standard Deviation
Spiral Path	625	-	-	-
GA	900	828	967	42.3
Basic ACO	711	653	761	29.5
Modified ACO (p = 1.0)	683	646	735	25.0
Modified ACO (p = 3.0)	659	613	656	13.2
Modified ACO (p = 5.0)	622	603	641	14.0
Modified ACO (p = 8.0)	608	594	630	12.4
Modified ACO (p = 12.0)	595	594	608	4.6
Modified ACO (p = 20.0)	594	594	594	0.0

Table 5. Results summary for Problem #2

Algorithm	Average Path Length (mm)	Min.	Max.	Standard Deviation
Spiral Path	944	-	-	-
GA	1434	1330	1509	62.1
Basic ACO	1144	1062	1280	67.2
Modified ACO (p = 1.0)	1021	928	1106	54.0
Modified ACO (p = 3.0)	946	922	980	20.2
Modified ACO (p = 5.0)	904	878	938	18.6
Modified ACO (p = 8.0)	833	813	859	15.6
Modified ACO (p = 12.0)	810	804	820	6.6
Modified ACO (p = 20.0)	804	804	810	2.0

Performance of the basic ACO was observed to be better than GA in both problems #1 and #2, however neither algorithm were capable of discovering paths with shorter travel distance than the simple spiral that would be generated via a typical CAD system (Figure 5, 6). On the other hand, the modified ACO with $p \geq 5$ was consistently capable of discovering paths that were better than the simple spiral. The best performance for the modified ACO was realized at $p = 20$, where the consistency in solution quality was very good (small value for standard deviation amongst the conducted runs) and the discovered paths (Figure 7) are deemed among the best-known for the problem. The fact that large values of p produce favorable results implies that "nearest neighbor" strategies are indeed favorable for this type of problem. However, due to the layout of the holes (many holes have comparable neighbor distances), favoring nearest-neighbor isn't realized except at larger values of p. It may be also worth noting the problems do have multiple equally-good solutions, but there is no mechanism to control which solution the ACO converges to. Plots in Figure 7 are simply those first discovered in the 10 runs.

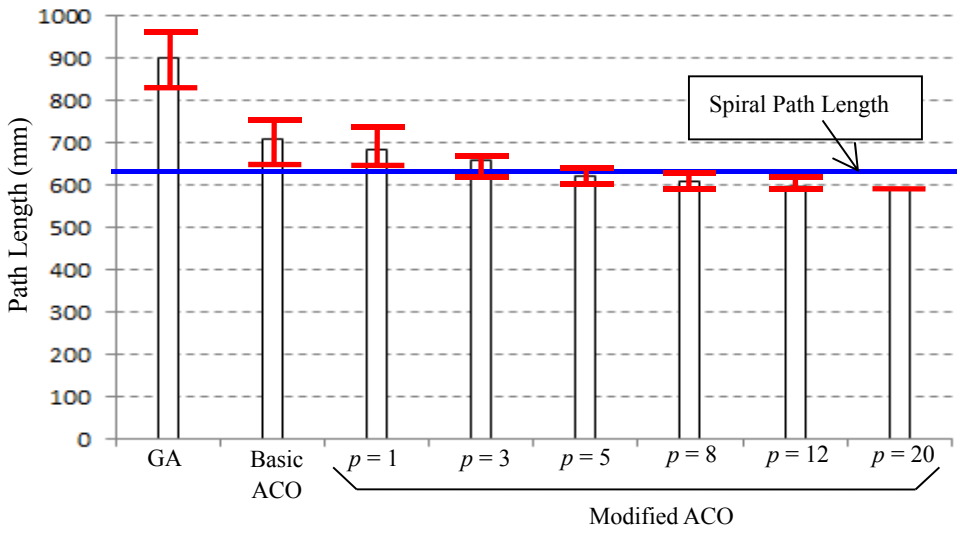

Figure 5. Results summary for problem #1

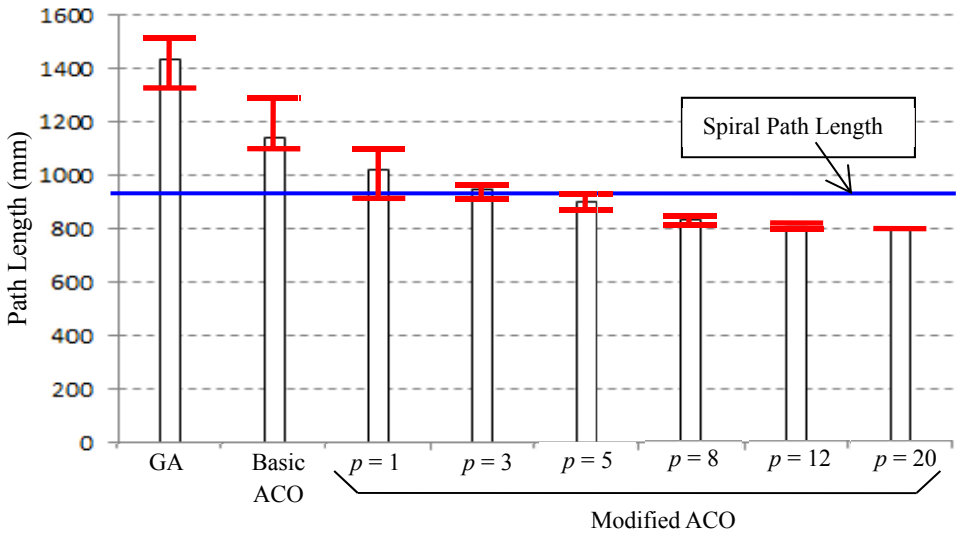

Figure 6. Results summary for problem #2

It is also worth noting that the solutions discovered via the modified ACO have 5% and 15% reduction in travel distance compared to the spiral path for problems #1 and #2 respectively. The travel distance savings via the proposed approach in problem #2 is observed to be more significant. This likely due to the increase in complexity of the geometric layout of the holes to be drilled, as opposed to problem #1 where the spiral path is already close to being optimal. This demonstrates both the capability and context of the proposed algorithm, where it becomes most beneficial when manufacturing similar complex products.

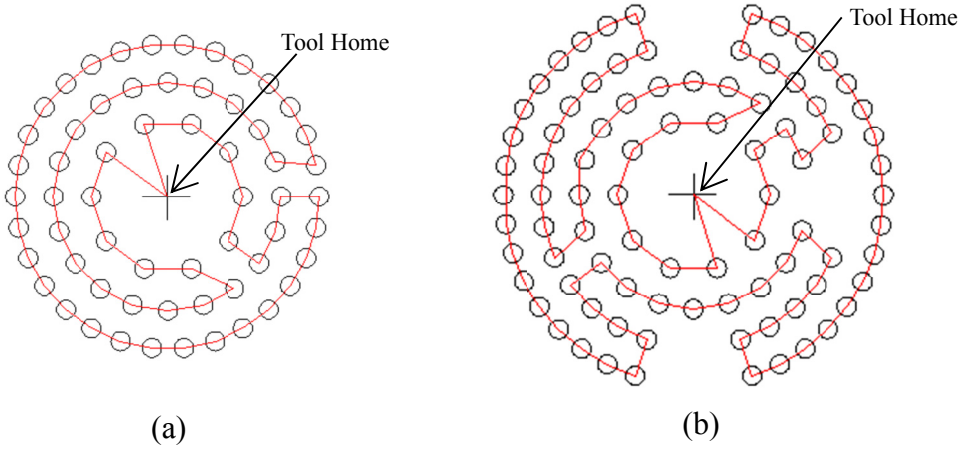

Figure 7. Selected best found paths for a) Problem #1, and b) Problem #2

5. Application Study

A full-scale food processing separator screen is chosen as an applications study to demonstrate the scalability of the proposed approach (Figure 1). A CAD model of the separator screen is shown in Figure 8 and its main geometrical parameters are listed in Table 6. The separator screen has a total of 2100 holes to be drilled (2101 nodes in the TSP), which is a very large sized problem from a TSP perspective. However, approaches derived from ACO generally scale well in terms of run-time. In this study, 5 runs of the modified ACO, with p = 20 were conducted on an Intel i3-2310M CPU @ 2.1 GHz, with all the algorithms implemented via in-house code on Matlab (MathWorks, 2010). The average time per run of the algorithm was approximately 90 min, which is a fairly

reasonable time-investment when planning a machining job. Summary of results of the conducted runs are listed in Table 7 and plots of the obtained paths are provided in Figure 9.

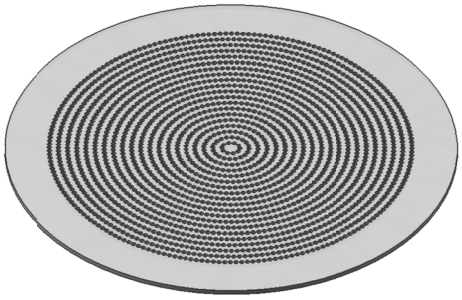

Figure 8. CAD model of separator screen

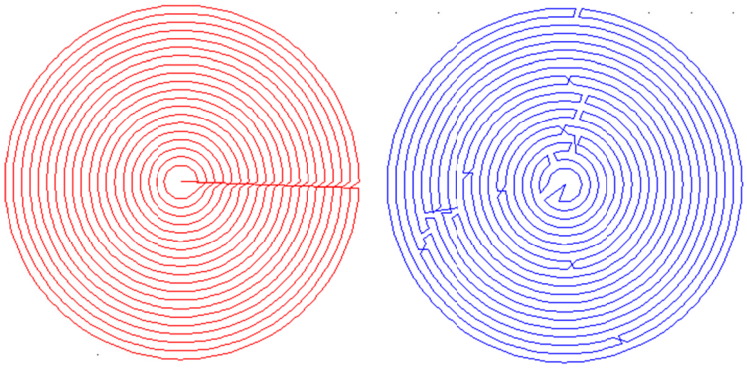

Figure 9. Tool paths for manufacturing the separator screen: a) Spiral path, and b) best found

Examining the best found tool-path (Figure 9.b), it is apparent that there are still some folds and crossovers that may be possible to improve upon. It is however noted that all five runs using the proposed approach (Table 7) were capable of discovering better solutions than the simple spiral path (99% confidence level using statistical *t*-test). This is despite the large size of the associated TSP and the fact that the spiral path is generally accepted as "close-to-optimum" by CAD systems and commercial tools for generating CNC G-code.

Table 6. Geometrical parameters of the Processing Separator

Parameter Description	Value
Hole diameter (mm)	5
Hole depth (mm)	10
Diameter of first pitch circle (mm)	40
Increment in pitch circle diameters (mm)	20
Number of holes in the first pitch circle	10
Increment in number of holes per pitch circle	10
Number of concentric pitch circles	20
Total number of holes	2100

Table 7. Results summary for industrial case study

Algorithm	Average Path Length (mm)	Min.	Max.	Standard Deviation
Spiral Path	14773	-	-	-
Modified ACO (p = 20.0)	14687	14655	14707	23.0

6. Conclusion

This paper presented a study on optimum path planning for drilling operation for products with large number of holes arranged in concentric circular patterns. The path planning is formulated as a traveling salesman problem. Solution of the TSP is tested via basic ant colony optimization and genetic algorithm, as well as a modified ACO where initialization of the pheromone matrix is biased towards traveling between nearby locations via a penalty function on the travel distance. Numerical simulations of example studies showed superior performance of the modified ACO algorithm compared to genetic algorithm and the basic ACO, with up to 15% reduction in total travel distance from the default spiral path in complex hole layouts. A model of a full-scale food separator screen with 2100 holes was used to demonstrate the scalability of the proposed approach.

Future extensions of this research may include application of the modified ACO approach to path planning in three-dimensional CNC tool-paths with axes re-orientation and/or tool changes.

Acknowledgments

This Project was supported by King Saud University, Deanship of Scientific Research, College of Engineering Research Center.

References

Aarts, E., & Korst, J. (1989). *Simulated annealing and Boltzmann machines: a stochastic approach to combinatorial optimization and neural computing*. New York: John Wiley & Sons Inc.

Abbas, A. T. (2012). Enhanced CNC machines capabilities by adding circular patterns cycle. *International Journal Precision Engineering and Manufacturing, 13*(10), 1753-1758. http://dx.doi.org/10.1007/s12541-012-0230-0

Abbas, A. T., & Megahed, S. M. (2005). A general algorithm for drilling holes lying in a matrix. *Robotics and Computer Integrated Manufacturing, 21*(3), 235-239. http://dx.doi.org/10.1016/j.rcim.2004.08.001

Abbas, A. T., Aly, M. F., & Hamza, K. T. (2011). Optimum drilling path planning for a rectangular matrix holes using ant colony optimization. *International Journal of Production Research, 49*(19), 5877-5891. http://dx.doi.org/10.1080/00207543.2010.507608

Anand, S., McCord, C., & Sharma, R. (1999). An integrated machine vision based system for solving the nonconvex cuttings stock problem using genetic algorithm. *Journal of Manufacturing Systems, 8*(6), 114-119

Arnaout, J. P., Rabadi, G, & Musa, R. (2010). A two-stage ant colony optimization algorithm to minimize the makespan on unrelated parallel machines with sequence-dependent setup times. *Journal of Intelligent Manufacturing, 21*, 693–701. http://dx.doi.org/10.1007/s10845-009-0246-1

Bonabeau, E., Dorigo M., & Theraulaz, G. (1999). *Swarm Intelligence: From Natural to Artificial Systems*. New York: Oxford University Press.

Denardo, E. V. (2004). Dynamic Programming: Models and Applications, Dover Publications

Dorigo, M. (1996). Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man and Cybernetics-Part B Cybernetics, 26*(1), 29-41. http://dx.doi.org/10.1109/3477.484436

Dorigo, M., & Stützle, T. (2004). Ant colony optimization. Cambridge, MA: MIT Press.

Dorigo, M., Birattari, M., Blum, C., Gambardella, L. M., Mondada, F., & Stuetzle, T. (2004). *Ant colony optimization and swarm intelligence*. 4th International Workshop, Springer-Verlag, Berlin Heidelberg

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

Ke, F., Ni, J., & Stephenson, D. A. (2006). Chip thickening in deep-hole drilling. *International Journal of Machine Tools and Manufacture, 46*(12-13), 1500-1507. http://dx.doi.org/10.1016/j.ijmachtools.2005.09.022

MathWorks. (2010). MATLAB Release 2010a. Natick, Massachusetts, United States: The MathWorks, Inc.

Michalewiz, Z. (1992). *Genetic algorithms + data structures = evolution programs* (2nd ed.). New York: Springer-Verlag.

Montiel-Ross, O., Medina-Rodríguez, N., Sepúlveda, R., & Melin, P. (2012). Methodology to Optimize Manufacturing Time for a CNC using a High Performance Implementation of ACO. *International Journal of Advanced Robotic Systems*. http://dx.doi.org/10.5772/50527

Padberg, M., & Sung T. (1991). An analytical comparison of different formulations of the travelling salesman problem. *Mathematical Programming, 52*, 315-357. http://dx.doi.org/10.1007/BF01582894

Saealal, M., Abidin, A., Adam, A., Mukred, J., Khalil, K., Yusof, Z., Ibrahim, Z., & Nordin, N. (2012). An ant colony system for routing in PCB holes drilling process. *International Journal of Innovative Management, Information & Production, 3*(1), 50-56

Satishkumar, S., & Asokan, P. (2008). Selection of optimal conditions for CNC multitool drilling system using non-traditional techniques. *International Journal of Machining and Machinability of Materials, 3*(1-2), 190-207. http://dx.doi.org/10.1504/IJMMM.2008.017633

Shunmugam, M. S., Bhaskara reddy, S. V., & Narendran, T. T. (2000). Optimal selection of parameters in multi tool drilling. *International Journal of Material Processing Technology, 103*, 318–323. http://dx.doi.org/10.1016/S0924-0136(00)00500-8

Stutzle, T., & Dorigo, M. (2002). A short convergence proof for a class of ant colony optimization algorithms. *IEEE Transactions on Evolutionary Computations, 6*(4), 358-365. http://dx.doi.org/10.1109/TEVC.2002.802444

Van Laarhoven, P., & Aarts, E. (2009). *Simulated annealing: theory and applications*. Berlin Heidelberg: Springer-Verlag.

Wolsey, L. A., & Nemhauser, G. L. (1999). *Integer and combinatorial optimization*. New York: John Wiley & Sons Inc.

Zhou, Y. (2009). Runtime analysis of an ant colony optimization algorithm for TSP Instances. *IEEE Transactions on Evolutionary Computations, 13*(5), 1083-1092. http://dx.doi.org/10.1109/TEVC.2009.2016570

Zhu, G. Y. (2006). Drilling path optimization based on swarm intelligent algorithm. *IEEE International Conference on Robotics and Biomimetics* (PP. 193-196). Kunming, China. http://dx.doi.org/10.1109/ROBIO.2006.340357

Zhu, G. Y., & Zhang, W. B. (2008). Drilling path optimization by the particle swarm optimization algorithm with global convergence characteristics. *International Journal of Production Research, 46*(8), 2299-2311. http://dx.doi.org/10.1080/00207540601042480