Tool Path Optimization for Computer Numerical Control Machines based on Parallel ACO

Nataly Medina-Rodríguez, Oscar Montiel-Ross, Roberto Sepúlveda, and Oscar Castillo

***Abstract*— In this paper, we present an efficient solution to determine the best sequence of G commands of a set of holes for a printed circuit board in order to find the hole-cutting sequence that shortens the cutting tool travel path. A Parallel proposal of Ant Colony Optimization was used to find an optimal travel path, then the new G-codes sequence is used instead the original sequence as part of the process program. This application can be formulated as a special case of the Traveling Salesman Problem (TSP).**

***Index Terms*— Ant Colony Optimization, ACO, Computer Numerical Control, CNC, Traveling Salesman Problem, TSP, drilling.**

I. Introduction

Computer Numerical Control (CNC) refers to the automation of machine tools, which is of primordial importance in any automated industrial process for manufacturing products. Today manual machine tools have been largely replaced by CNC machines where all movements of the machine tools are programmed and controlled electronically rather than by hand [15], reducing time and avoiding human errors. The productivity of CNC machine tools is significantly improved by using Computer Aided Design (CAD) and Computer Aided Manufacturing (CAM) systems for automated Numerical Control (NC) program generation. Currently, many CAD/CAM packages that provide automatic NC programming have been developed for various cutting processes, being one of those process the hole – cutting operation or drilling.

There are several studies that focus on the study of reducing the cutting time by optimizing some parameters such as part geometry, material and tool type. This paper analyzes the cutting time, which is the time that the cutting tool moves with cutting speed in air or in material. A survey of the literature shows that much research has been done on minimizing the cutting time [1, 2]; however, there is a lack of literature that studies the travel time between operations. In order to minimize the travel time, the cutting tool travel path between operations should be minimized.

Manuscript received July 10, 2011; and accepted August 15, 2011.

Nataly Medina Rodríguez is a PhD student in Intelligent Systems in Instituto Politécnico Nacional - CITEDI. Av. del Parque 1310, Otay, Tijuana, B.C., México. (e-mail: nmedina@citedi.mx).

Oscar H. Montiel Ross is with Instituto Politécnico Nacional - CITEDI. (Corresponding author phone: 52(664)-623-1344;
e-mail: o.montiel@ieee.org.)

Roberto Sepúlveda is with Instituto Politécnico Nacional - CITEDI.
e-mail: r.sepulveda@ieee.org.

Oscar Castillo is with Tijuana Institute of Technology in the Department of Computer Science in the Graduate Division. Calzada Tecnológico S/N, Tijuana, B. C., México. (e-mail ocastillo@hafsamx.org).

This travel path can be formulated as a special case of the traveling salesman problem (TSP) [3].

II. Ant Colony Optimization for the Traveling Salesman Problem

The TSP problem [13][14] is the problem of a salesman who, starting from his hometown, wants to find a shortest tour that takes him through a given set of costumer cities and then back home, visiting each customer city exactly once. The TSP can be represented by a complete weighted graph [4] $G = (N, A)$ with N being the set of nodes representing the cities, and A being the set of arcs. Each arc $(i, j) \in A$ has assigned a value (length) d_{ij}, which is the distance between cities *i* and *j*.

Ant Colony Optimization was introduced by Marco Dorigo [4]. Using very simple communication mechanisms, an ant group is able to find the shortest path between any two points by choosing the paths according to pheromone levels.

After several years that ACO was introduced, many papers have described applications that use this algorithm.

For example in robotic, in [16] the Simple ACO was applied to obtain the optimal path for a mobile robot, here was considered static and dynamic obstacle avoidance, a memory capacity for the ants was proposed, and a fuzzy cost function was used. In [17] and [18] the ACO was applied to tune fuzzy parameters of a fuzzy logic controller for a wheeled mobile robot, in [19] a comparison of ACO and Genetic Algorithms applied to fuzzy system optimization was presented.

ACO metaheuristics can be applied to the TSP, where the pheromone trails are associated with arcs and therefore τ_{ij} refers to the desirability of visiting city *j* directly after city *i*. The heuristic information is chosen as $\eta_{ij} = 1/d_{ij}$; that is, the heuristic desirability of going from city *i* to city *j* is inversely proportional to the distance between the two cities. For implementation purposes, pheromone trails are collected into a pheromone matrix whose elements are the τ_{ij}'s.

Tours are constructed by applying the following simple constructive procedure to each ant:

1. Each ant chooses, according to some criterion, a start city at which the ant is positioned.
2. Each ant uses a pheromone and heuristic values to probabilistically construct a tour by iteratively adding cities that the ant has not visited yet, until all cities have been visited.
3. Each ant goes back to the initial city.
4. After all ants have completed their tour, they may deposit pheromone on the tours they have followed.

In some cases, before adding pheromone, the tours constructed by ants may be improved by the application of a local search procedure. The above procedure can be represented by the following pseudocode showed in Fig. 1:

```
procedure ACO Metaheuristic
   Set parameters, initialize pheromone trails
   while (termination condition not met) do
         Construct Ant Solutions
         Apply Local Search
         Update Pheromones
  end
end
```

Fig. 1 Tours are constructed by applying this simple constructive procedure to each ant: ACO Metaheuristic.

A. *Ant System*

There are two main phases in the Ant System (AS) algorithm [12]; these are the ants' solution construction and the pheromone update. In AS a good heuristic to initialize the pheromone trails is to set them to a value slightly higher than the expected amount of pheromone deposited by the ants in one iteration; a rough estimate of this value can be obtained by setting, $\forall(i,j), \tau_{ij} = \tau_0 = m/C^{nm}$, where *m* is the number of ants, and C^{nm} is the length of a tour generated by the nearest-neighbor heuristic.

B. *Tour Construction*

In AS, *m* ants concurrently build a tour of the TSP. Initially, ants are put on randomly chosen cities. At each construction step, ant *k* applies a probabilistic action choice rule, called random proportional rule, to decide which city is going to visit next.

The probability with which ant *k*, currently at city *i*, chooses to go to city *j* is (1):

$$p_{ij}^k = \frac{[\tau_{ij}]^\alpha [\eta_{ij}]^\beta}{\sum_{l \in N_i^k} [\tau_{ij}]^\alpha [\eta_{ij}]^\beta}, \quad if \quad j \in N_i^k \tag{1}$$

Where $\eta_{ij} = 1/d_{ij}$ is a heuristic value, α and β are two parameters which determine the relative influence of the pheromone trail and the heuristic information, and N_i^k is the feasible neighborhood of ant *k* when being at city *i*, that is, the set of cities that ant *k* has not visited yet. The probability of choosing a city outside N_i^k is 0. The role of the parameters α and βcan be set by the following statements:

- If $\alpha = 0$, the closest cities are more likely to be selected (α is parameter to regulate the influence of τ_{ij}).
- If $\beta = 0$, only pheromone amplification is used, without any heuristic bias ($\beta$is parameter to regulate the influence of η_{ij}).

C. *Update of pheromone trails*

After all the ants have constructed their tours, the pheromone trails are updated. This is done by first lowering the pheromone value on all arcs by a constant factor, and then adding pheromone on the arcs the ants have crossed in their tours.

Pheromone evaporation is implemented by (2):

$$\tau_{ij} \leftarrow (1-\rho)\tau_{ij}, \quad \forall(i,j) \in L, \tag{2}$$

Where $0 \leq \rho \leq 1$ is the pheromone evaporation rate. After evaporation, all ants deposit pheromone on the arcs they have crossed in their as shown in (3):

$$\tau_{ij} \leftarrow \tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^k, \quad \forall(i,j) \in L, \tag{3}$$

Where Δ_{ij}^k is the amount of pheromone ant *k* deposits on the arcs it has visited it is defined by (4):

$$\Delta\tau_{ij}^k = \begin{cases} \frac{1}{C^k}, & \text{if arc } (i,j) \text{ belongs to } T^k \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

Where C^k is the length of the tour T^k built by the *k*-th ant, is computed as the sum of the lengths of the arcs belonging to T^k.

III. Parallel Implementation of Ant Colony Optimization

The ACO algorithm can be formulated by the following sequential implementation as shown in Fig. 2:

```
procedure ACO Sequential Metaheuristic
   begin
   Initialize Parameters
 for each cycle
      for each ant
         for each city
             Build a solution k
             Evaluate solution k
           end for
         end for
         Save Best Solution
         Update Trail
      end for
   Print Best Solution
   end
```

Fig. 2 Ant Colony Optimization extended sequential metaheuristic.

The availability of parallel architectures at low cost has widened the interest for the parallelization of Ant Colony Optimization algorithm. In order to parallelize the ACO, it is more important to modify the structure of ACO to get better optimization effect rather than to transfer the sequential ACO into a parallelization schema [5].

The main purpose of parallel implementation of ACO is to obtain the high speedup and efficiency while the convergence and the ability of optimization are maintained or even improved.

Some results on parallel Ant Colony algorithms have been reported recently. Bullnheimer [6] proposed two parallelization strategies of synchronous and asynchronous for ACO using the Traveling Salesman Problem. Piriyakumar in [7] introduced an asynchronous parallel Max-Min ACO associated with the local search strategy. Marcus Randal in [8] introduced a synchronous parallel strategy which assigns only one ant on each processor. Marco Dorigo in [9] introduced a parallel ACO on the hyper – cube architecture by modifying the rule of updating the pheromone so as to limit the pheromone values within the range of [0,1].

In this paper, we present a parallel implementation of ACO for CNC Tool path optimization generating a set of G commands. The general procedure consists on the interpretation of a DXF file as input, detecting commands related to the coordinates of all points; then this information is optimized by using Ant Colony Optimization and finally, our system generates the best route found by ants so we interpret this route by generating a set of G commands.

IV. Parallel Implementation: Problem Formulation

The sequential algorithm contains a high degree of natural parallelism [6], the behavior of a single ant during one iteration is totally independent of the behavior of all other ants during that iteration. We will discuss a strategy called “synchronous (fork – join) algorithm”.

A straight forward parallelization strategy for ACO is to compute the TSP tours in parallel; this would result in a fork – join structure as shown in Fig. 3.

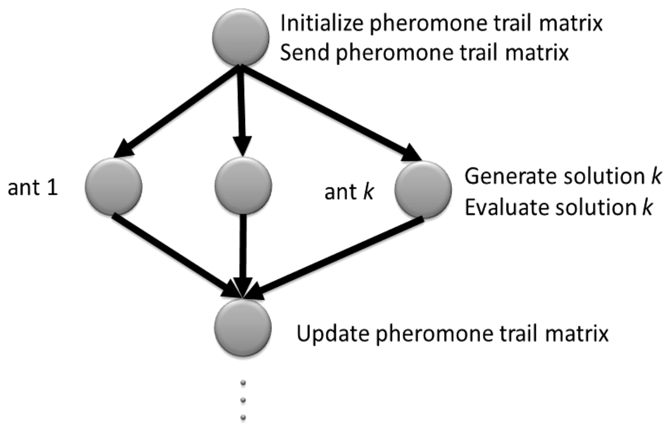

Initialize pheromone trail matrix
Send pheromone trail matrix
ant 1
ant *k*
Generate solution *k*
Evaluate solution *k*
Update pheromone trail matrix

Fig. 3 Parallel implementation of Ant Colony Optimization in a message passing model.

An initial process (master) would spawn a set of processes, one for each ant. After distributing initial information about the problem, each process can generate a single solution for each ant *k*. After finishing this procedure, the result is sent from each process back to the master process. The master process updates the trail levels by calculating the intensity of the trails and checks for the best tour found so far. A new iteration is initiated by sending out the updated trail levels.

Bullnheimer [6] says that ignoring any communication overhead, this approach would imply optimum (asymptotic) speedup as (5), assuming that an infinite number of processing elements (workers) is available, i.e., one process is assigned to one worker. In (5) *m* is the quantity of ants of the colony.

$$S_{asymptotic}(m) = \frac{T_{sec}(m)}{T_{par}(m,\infty)} = \frac{\mathrm{O}(m^3)}{\mathrm{O}(m^2)} = \mathrm{O}(m) \qquad (5)$$

Where $T_{seq}(m) = O(m^3)$ the computational complexity of the sequential algorithm for problem is size m and $T_{par}(m,\infty) = O(m^2)$ is the computational complexity of the parallel algorithm for problem size *m* and for infinite system size.

Communication overhead certainly cannot be disregarded and has to be taken into account, and further the system size (number of processing elements *N*) is restricted and is typically smaller than the problem size (number of ants *m).*

Balancing the load among the workers is easily accomplished by assigning to worker $(j = 1...N)$ the processes (ants) m_i for $(i = 1...m)$ according to $m_i : j = i \ mod \ N$, thus each worker holds about the same number of processes and each process is of the same computational complexity [6].

When considering communication overhead, the ratio of the amount of computation assigned to a worker and the amount of data to be communicated has to be balanced [6].

After each iteration, all completed tours and their lengths have to be sent to a central process (master). Then the new trail levels need to be computed and then broadcasted to each worker which only then can start a new iteration.

A. *Parallel Ant Colony Algorithm Framework*

In our parallel ACO, the ants are divided equally into *P* groups which are allocated into *P* processors. The ants in each group search for the best solution in its own processor independently as shown in Fig. 4.

```
begin
  An initial process initializes the pheromone
matrix and some other control parameters.
  while (not terminate) do
    for each processor do in parallel
      for each ant do
        Build a solution k
        Evaluate solution k
      end for
    end for
```

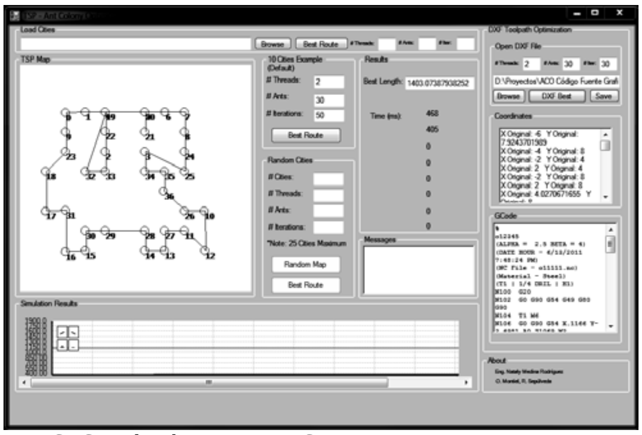
Fig. 4 Parallel Ant Colony Optimization framework.

The main part of the algorithm is complexity $O(n^3)$ and the generation of one solution is complexity $O(n^2)$ where *n* is the number of jobs; these two operations are independent for each ant of a given cycle so they can be easily parallelized.

Fig. 3 shows the behavior of our implementation of the ACO based on the parallel synchronous Ant System in a message passing model. At the beginning of the algorithm, a master process initializes the information, spawns *k* processes (one for each ant *k*), and broadcasts the information. At the beginning of a cycle the τ_{ij} matrix (the pheromone trail) is sent to each process and the computations such generation and evaluation of solutions, are done in parallel. Then, the solutions and their evaluations are sent back to the master, the τ_{ij} matrix is updated and a new cycle begins by the broadcasting of the updated τ_{ij} matrix.

V. CNC Tool Path Generation

The TSP and Parallel ACO have been incorporated to find the shortest cutting tool travel path to operate the holes in the next figures. Fig. 5 shows our GUI developed with .NET Framework.



Threads: 2
Ants: 30
Iterations: 50
Best Length: 1403.07387908252
Time (ms): 468 405 0 0 0 0 0 0
*Note: 25 Cities Maximum
D:\Proyectos\ACO Código Fuente Grafi

```
X Original: -6  Y Original: 7.9243701589
X Original: -4  Y Original: 8
X Original: -2  Y Original: 4
X Original: 2  Y Original: 4
X Original: -2  Y Original: 8
X Original: 2  Y Original: 8
X Original: 4.0270671655  Y
```

```
%
o12345
(ALPHA =  2.5 BETA = 4)
(DATE HOUR - 4/13/2011
7:48:24 PM)
(NC File - o11111.nc)
(Material - Steel)
(T1 | 1/4 DRIL | H1)
N100  G20
N102  G0 G90 G54 G49 G80
G90
N104  T1 M6
N106  G0 G90 G54 X.1166 Y-
```

Eng. Nataly Medina Rodriguez
O. Montiel, R. Sepúlveda

Fig. 5 CNC Tool path optimization GUI.

A number of commercial CAD/CAD packages that provide automatic NC programming have been developed and applied to various cutting processes. To cut any process using CNC machine tools, a tool path for the cutting tool should be determined [3].

The total production time to cut any part using CNC machine tools consist of *travel time* [3], which is the time to move the CNC machine spindle between operations, *switch time* which is the time to change the cutting tool for next operation and the *cutting time*, the time that the cutting tool moves with cutting speed in air or in material. Our purpose is minimize the cutting time based on Parallel Ant Colony Optimization.

In order to minimize the cutting time, the cutting tool travel path (CTTP) [3] between operations should be minimized. We were working on a continuous travel path, in which the start point and the end point of each operation are the same, and it mainly appears in hole – cutting operations such as drilling, reaming and tapping.

A. *System Development: Tool path programming optimization strategy.*

To exchange files between CAD/CAM packages, it is necessary a translator, which takes place through intermediate files, as show in Fig. 6. These files can be any of the following standard formats: Drawing Exchange Files (DXF), IGES files and STEP files [10].

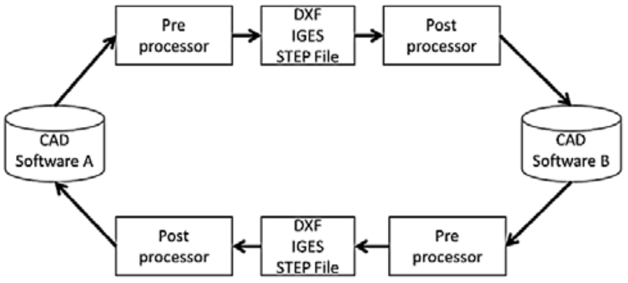


Fig. 6 Exchange files between CAD/CAM packages.

Our system reads an input DXF file to obtain a set of commands and coordinates for any part designed in a CAD/CAM commercial package; this input DXF file is now being processed by the Parallel Ant Colony Optimization algorithm to generate a sequence of G commands for a CNC drilling tool path. This implementation can be represented as shown in Fig 7.

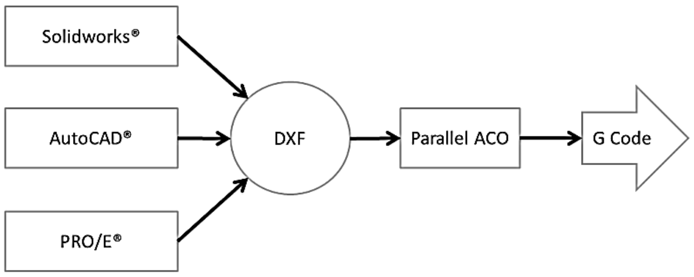


Fig. 7 CNC Tool path optimization for a drilling process.

B. *DXF file interpretation*

For an input DXF file, i.e., as shown in Fig 8. a translation procedure must be done.

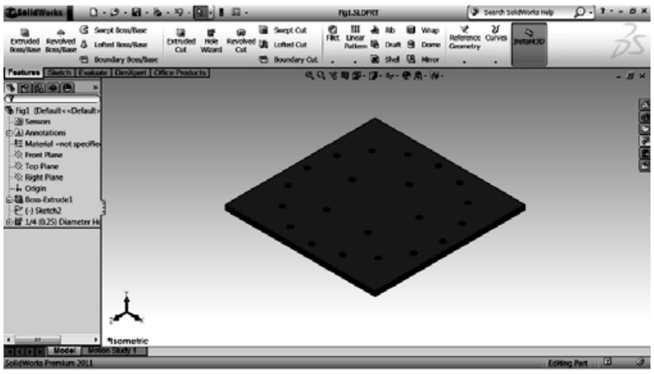


Fig. 8 3D CAD drawing.

In Fig. 9 a script containing a set of DXF commands that represent a single hole of Fig. 8 is shown. This hole is located at X = 1.06516 and Y = -1.5176 with a diameter of 1/8” in. The interpretation procedure can be represented by the pseudocode showed in Fig. 10.

```
0                  Continuous
SECTION            62
2                  7
ENTITIES           370
0                  15
CIRCLE             100
5                  AcDbCircle
48                 10
330                -1.0651610148
1F                 20
100                -1.5176053881
AcDbEntity         30
8                  0.0
0                  40
```

Fig. 9 DXF script describing a hole and its location.

```
reader = ReadFromFile(file.dxf)

  while(line2 ¡= "EOF")
    if line1 = 0 and line2 = "LINE"
      LineModule(reader);
    else if line1 = 0 and line2 = "CIRCLE"
      CircleModule(reader);
GetLineCouple();
```

Fig. 10 DXF interpretation procedure.

C. G code generation for the Cutting Tool Travel Path

The following procedure represents the generation of G code sequence for the cutting tool travel path:

- *Step 1*. Read the coordinate of each node of the optimized CTTP based on Parallel ACO.
- *Step 2*. Code the traverse motion command G00 and then the X and Y coordinates of the first hole in the CTTP.
- *Step 3*. For each coordinate in solution *k*, code G00 rapid move and then X and Y coordinates for the next node.
- *Step 4*. If a change in tool is needed, then code M6 for a tool change and repeat steps 1 to 3. If no change in tool is needed, then proceed with steps 1 to 3 until the cutting tool reaches the last hole in the cutting travel tool path.

Our system generates this output file and then the user can save this file with .NC extension.

VI. Experimental Results and Analysis

In this section, we show the test results of our parallel ACO algorithm on some PCB boards with a different number of holes. The parameters in the test are set as follows:

$\rho = 0.1$ Evaporation coefficient.
$\alpha = 1,\ 2,\ 2.5$ Parameter to regulate the influence of τ_{ij}.
$\alpha = 4,\ 4.5$ Parameter to regulate the influence of η_{ij}.

The number of ants is equal to the number of holes and the number of processors is 6. Our experiment performs 50 trials on each problem. The experimental results are shown in Figures 12, 14, and 15, where three TSP problems are tested based on Parallel ACO and they are compared with a classical ACO algorithm by using a single processor.

A. Parallel Implementation of ACO Analysis

1) Experiment I. PCB with 10 holes.
Table I shows the parameters values used in the first experiment which consists of a PCB board with 10 holes. Fig. 11 shows the first experiment, generating the optimized tool path and Fig. 12 shows a graph representing these experimental results. We can observe that the convergence point for three different values of α and β is given by using more than four processors while by using less than four processors we can observe an average difference of 30mS. After these points, further iterations do not provide significant improvements in the execution time, considering that we have only used six processors for all these three experiments.

TABLE I
PARAMETERS FOR EXPERIMENT I

Description	I-A	I-B	I-C
Num. Holes	10	10	10
Num. Ants	30	30	30
Iterations	30	30	30
Alpha	1	2	2.5
Beta	4	4.5	4

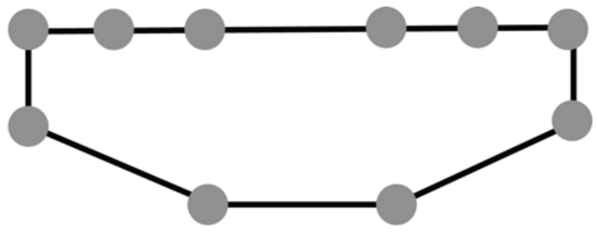


Fig. 11 Graphical results for experiment 1.

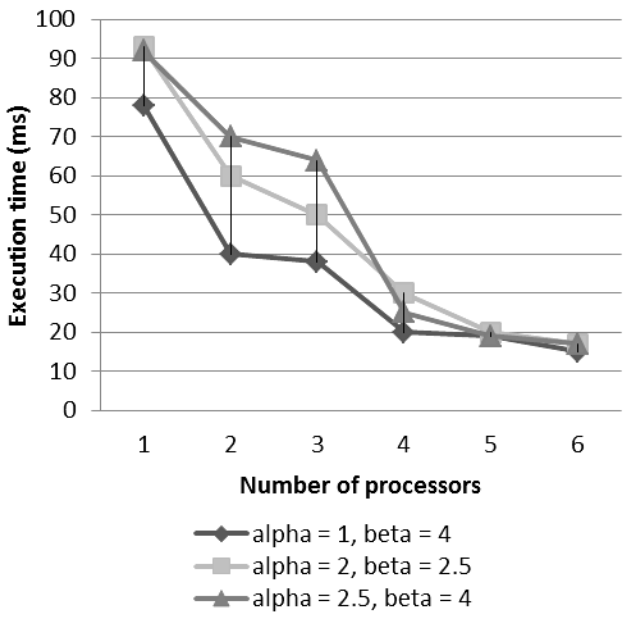


Fig. 12 Execution time according to the number of processors in experiment 1.

2) Experiment II. PCB with 27 holes.
Table II shows the parameters values used in the second experiment consisting in the optimization of the tool path of a PCB board with 27 holes, and Fig. 13 shows the second experiment, generating the optimizing tool path; Fig. 14 shows a graph representing these experimental results.

TABLE II
PARAMETERS FOR EXPERIMENT II

Description	2 – A	2 – B	2 – C
Num. Holes	27	27	27
Num. Ants	30	30	30
Iterations	30	30	30
Alpha	1	2	2.5
Beta	4	4.5	4

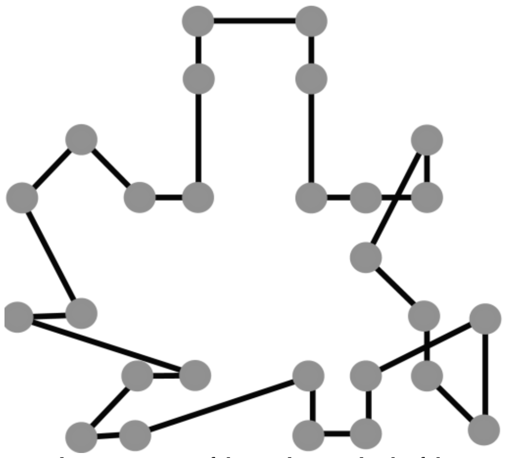


Fig. 13. Visual representation of the resulting toolpath of the experiment 2.

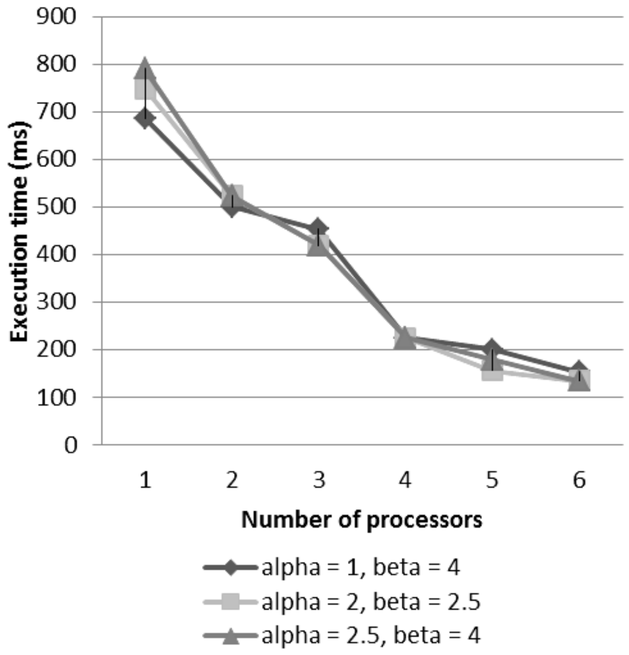


Fig. 14. Execution time according to the number of processors in experiment 2.

In this experiment, we can observe that there is no difference using these three different values for α and β; however, the execution time consistently decreases as the number of processors used increases.

3) Experiment III. PCB with 45 holes.

Table III shows the set of parameters that were used in the third experiment which consists of a PCB board with 45 holes. The execution time according to the number of processors for this experiment is showed in Fig. 15.

TABLE III
PARAMETERS FOR EXPERIMENT III

Description	3 – A	3 – B	3 – C
Num. Holes	45	45	45
Num. Ants	70	70	70
Iterations	70	70	70
Alpha	1	2	2.5
Beta	4	4.5	4

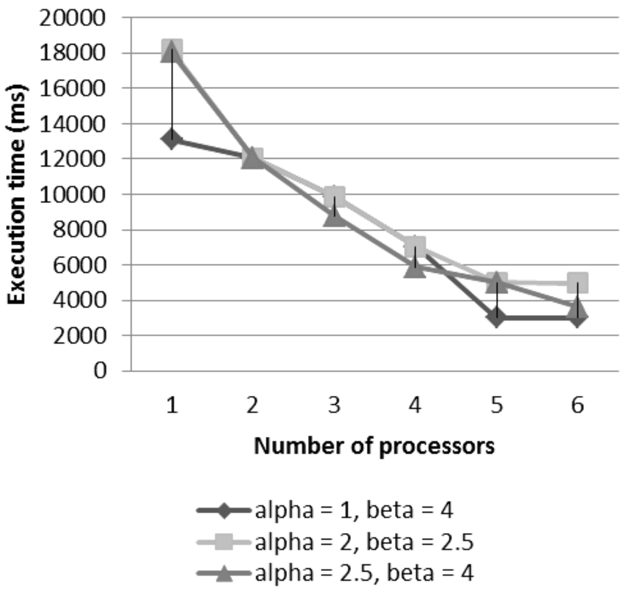


Fig. 15. Execution time according to the number of processors in experiment 3.

In experiment III, we can observe a similar trend line as experiment II, but we note that we have less execution time difference between these three cases than in experiment I. We can observe that for a 45 holes PCB, α and β can still being the same as in experiment II.

It can easily be seen from Figures 12, 14 and 15 that Parallel ACO computation is reduced due the parallel computation. The reason for Parallel ACO's high optimization ability is that it can accelerate the convergence by dividing the ants into smaller groups allocated in the processors.

We can observe that when the number of processors is increased, the computing time can be reduced due to the fewer ants assigned on each processor. But due to the overhead of communication which increases the total time of algorithm, the speedup of our algorithm cannot increase linearly with the increasing of processor exactly, but there is an excellent linear trend as shown in Fig. 16. This is in conformity with the Amdahl's Law [11].

In experiment 1 we can see that a difference occurs until three processors are used, more than three processors may not present a real difference between experiments. In experiment 2 we can see that there is no difference between experiments by changing the control parameters, α and β.

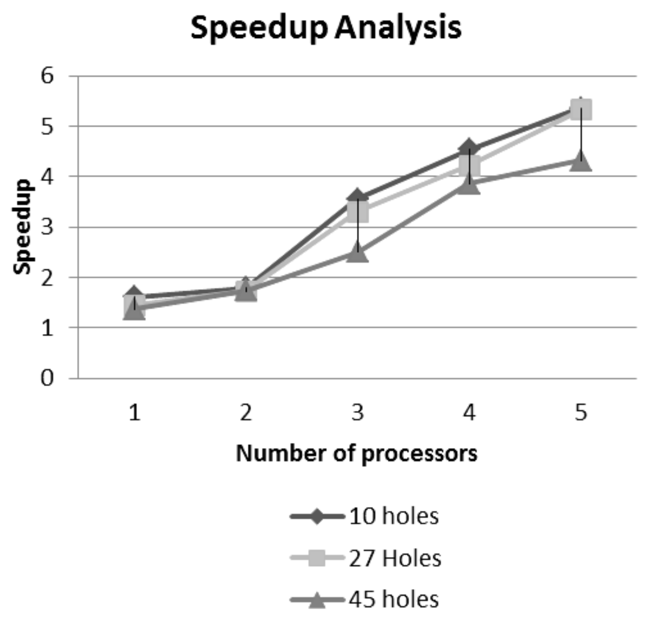


Fig. 16. Speedup analysis.

B. Tool path optimization analysis

For a tool path optimization analysis, we present two cases in which the PCB board area is very important; this means that for a larger area, there is more difference on manufacturing times. Figure 17 shows a HAAS Automation® CNC machine used for our experiments. The main idea is to compare our results against those obtained using commercial software such as MasterCam®.

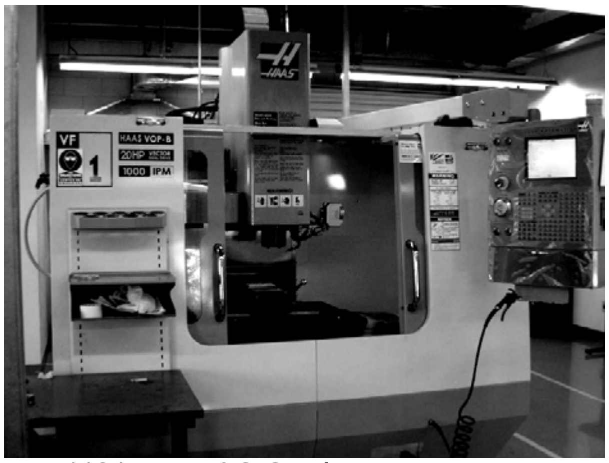


Fig. 17. HAAS Automation® CNC Machine.

The three experiments were achieved using two different sizes of PCBs in order to use different lengths between holes; so, we have the case A that consist of a PCB with an area of $64\ in^2$, and the case B is a PCB with an area of $225\ in^2$. For the case A, Fig. 18 shows a plot of cutting time vs. number of holes. We can see that for 10 holes, there is a difference of 1.6 minutes between a route generated by a commercial package and our implemented algorithm; furthermore, for a 35 holes to 45 holes PCB we can see a reduction in the difference between these two algorithms.
In Fig. 19 the case B is shown, in the plot can be observed that for a bigger area, more difference between these two CAD/CAM systems for automatic NC programming.

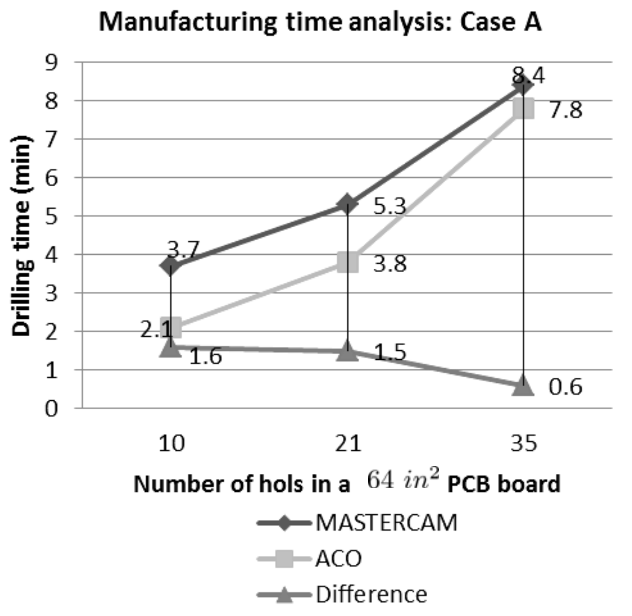


Fig. 18. Manufacturing time analysis in case A.

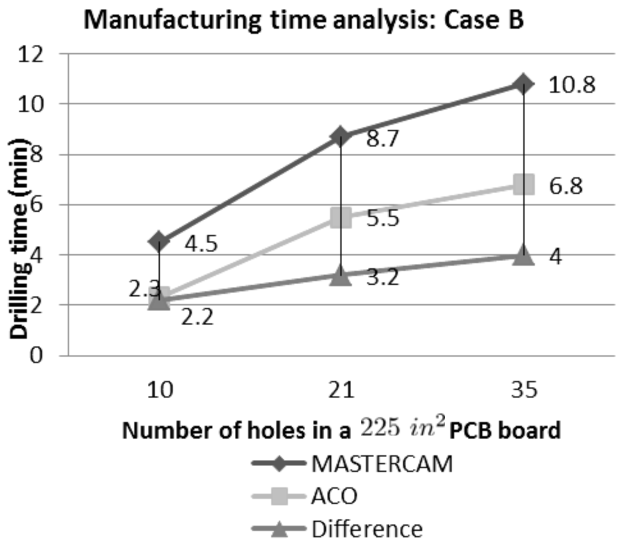


Fig. 19. Manufacturing time analysis in case B.

VII. Conclusion

In this paper we presented a parallel implementation of Ant Colony Optimization, to improve the efficiency by means of increasing the speed up. The availability of parallel architectures at low cost has widened the interest for the parallelization of ACO algorithm.

Furthermore, the development of computer aided machine tool programming has facilitated a great reduction in manufacturing times. This paper finds an efficient sequence of operation for a set of holes located in a PCB board that achieves the shortest tool path. This path was formulated as an application of the TSP. The incorporation of an algorithm based on Parallel Ant Colony Optimization and TSP can be applied to any similar problem, such welding and tapping, and this can be included in commercial CAD/CAM packages to optimize these tool paths during automatic generation of CNC machine programs.

ACKNOWLEDGEMENT

We would like to thank CETYS Universidad campus Tijuana for supporting this research, providing access to manufacturing equipment specifically a HAAS Automation® CNC machine and training.

REFERENCES

[1] K. Castelino, P. K. Wright., "Tool path optimization for minimizing airtime during machining", Journal of Manufacturing Systems, Vol. 22, No. 3, pp. 173-180, 2003.

[2] F. Kolahan., and M. Liang, "Optimization of hole-making operations: a Tabu-search approach", *International Journal of Machine Tools & Manufacture,* Vol.40, No. 12, pp. 1735-1753, 2000.

[3] J. E. A. Qudeiri, Al-Momani Raid, Mohamed Anouar Jamali and Hidehiko Yamamoto, "Optimization Hole-cutting Operations Sequence in CNC Machine Tools Using GA". *International Conference on Service Systems and Service Management* (ICSSSM06), Troyes, France, 2006, pp.:501,506.

[4] M. Dorigo, T. Stützle. *Ant Colony Optimization.* MIT Press, Cambridge, MA. 2004.

[5] L. Chen, Hai-Ying Sun, Shu Wang. "Parallel Implementation of Ant Colony Optimization on MPP". *Proceedings of the Seventh International Conference on Machine Learning and Cybernetic*s Kunming: 2008

[6] B. Bullnheimer, G. Kotsis, C. Steauss, "Parallelization strategies for the ant system". *High Performance and Algorithms and Software in Nonlinear Optimization, Applied Optimization*: 1998.

[7] Douglas Antony Louis Piriyakumar, Paul Levi. "A new approach to exploiting parallelism in Ant Colony Optimization". *Proceedings of 2002 International Symposium on Micromechatronics and Human Science*: 2002.

[8] M. Randall. "A parallel implementation of Ant Colony Optimization". *Parallel and Distributed Computing*: 2002.

[9] C. Blum, M. Dorigo. "The Hyper – Cube Framework for Ant Colony Optimization". *IEEE Transactions on SMC*: 2004.

[10] P. Radhakrishnan, S. Subramanyan, V. Raju. "CAD/CAM/CIM". New Age International Publishers, 2000.

[11] L. Null, J. Lobur. *The essentials of computer organization and architecture.* Jones and Barlett Publishers. 2006.

[12] M. Dorigo, V. Maniezzo, A. Colomi, "Ant System: Optimization by a Colony of Cooperating Agents." *IEEE Transactions on Systems, Man and Cybernetics-Part B.* 1996.

[13] M. Dorigo, L.M. Gambardella., "Ant Colony System: a cooperative learning approach to the traveling salesman problema." *IEEE Trans. On Evolutionary Computation*. 1997.

[14] M. Dorigo, L.M. Gambardella, "Ant colonies for the traveling salesman problem". *BioSystems*. 1997.

[15] M. Mattson. "CNC Programming: Principles and Applications." Delmar Cengage Learning. 2010.

[16] M. A. Porta García, O. Montiel, O. Castillo, R. Sepúlveda, P. Melin, "Path planning for autonomous mobile robot navigation with ant colony optimization and fuzzy cost function evaluation." *Appl. Soft Comput*. 9(3): 1102-1110 (2009)

[17] O. Castillo, R. Martinez-Marroquin, J. Soria, "Parameter Tuning of Membership Functions of a Fuzzy Logic Controller for an Autonomous Wheeled Mobile Robot Using Ant Colony Optimization". *SMC* 2009: 4770-4775

[18] R. Martínez-Soto, O. Castillo, J. Soria: "Optimization of Membership Functions of a Fuzzy Logic Controller for an Autonomous Wheeled Mobile Robot Using Ant Colony Optimization." *Evolutionary Design of Intelligent Systems in Modeling, Simulation and Control 2009*: pp. 3-16, 2009

[19] A. C. Martinez, O. Castillo, O. Montiel, "Comparison between Ant Colony and Genetic Algorithms for Fuzzy System Optimization." *Soft Computing for Hybrid Intelligent Systems 2008*: pp. 71-86, 2008

Nataly Medina Rodríguez. She received his Engineering title in Cybernetic Electronics from Centro de Enseñanza Técnica y Superior (CETYS) and she obtained her M. Sc. Degree from the Instituto Politécnico Nacional – CITEDI. Currently she is a professor at CETYS Universidad campus Tijuana and a Ph.D. student in Computer Science in the field of Intelligent Systems at Instituto Politécnico Nacional. She has published papers about Ant Colony Optimization applied to combinatorial optimization problems and robotics. Her research interests include optimization, intelligent systems, robotics, neuroscience, human interface devices, computer graphic, software design and electronics.

Oscar Humberto Montiel Ross. He received the M. Sc. From the Tijuana Institute of Technology, Tijuana, México, and the Ph. D. degree in the Universidad Autónoma of Baja California, Tijuana, México, both in Computer Science, in 2000 and 2006 respectively. He has published papers about evolutionary computation, Mediative Fuzzy Logic, Ant Colonies, type-2 fuzzy systems, embedded systems, and mobile robotic. He works as a researcher at the Centro de Investigación y Desarrollo de Tecnología Digital (CITEDI) of the Instituto Politécnico Nacional. His research interests include optimization, intelligent systems, and robotics. Dr. Montiel is co-founder and an active member of HAFSA (Hispanic American Fuzzy Systems Association), and the Mexican Chapter of the Computational Intelligence Society (IEEE), he is member of the International Association of Engineers (IANG).

Roberto Sepúlveda Cruz. He received the M. Sc. from the Tijuana Institute of Technology, Tijuana,México, and the Ph. D. degree in the Universidad Autónoma of Baja California, Tijuana, México, both in Computer Science, in 1999 and 2006 respectively. He has published papers about type-2 fuzzy systems, embedded type-1 and type-2 fuzzy logic controllers, mobile robot navigation. He works as a researcher at the Centro de Investigación y Desarrollo de Tecnología Digital of the Instituto Politécnico Nacional. His research interests include type-2 fuzzy systems, intelligent systems, and robotics. Dr. Sepúlveda is co-founder and an active member of HAFSA (Hispanic American Fuzzy Systems Association), and the Mexican Chapter of the Computational Intelligence Society (IEEE), he is member of the International Association of Engineers (IANG).

Oscar Castillo. He holds the Doctor in Science degree (Doctor Habilitatus) in Computer Science from the Polish Academy of Sciences (with the Dissertation "Soft Computing and Fractal Theory for Intelligent Manufacturing"). He has published over 130 research papers, u authored books, 10 edited books, and 200 papers in conferences proceedings about Soft Computing. He is a Professor of Computer Science in the Graduate Division, Tijuana Institute of Technology, Tijuana, México. In addition, he is serving as Research Director of Computer Science and head of the research group on fuzzy logic and genetic algorithms. President of IFSA (International Fuzzy Systems Association); he is also Vice-Chair of the Mexican Chapter of the Computational Intelligence Society (IEEE).