Methodology to Optimize Manufacturing Time for a CNC Using a High Performance Implementation of ACO

Regular Paper

Oscar Montiel-Ross1,*, Nataly Medina-Rodríguez1, Roberto Sepúlveda1 and Patricia Melin2

1 Instituto Politécnico Nacional - CITEDI, Av. del Parque 1310, Tijuana, B.C., México
2 Department of Computer Science, Tijuana Institute of Technology, México
* Corresponding author E-mail: o.montiel@ieee.org

Received 22 Apr 2012; Accepted 7 Jun 2012
DOI: 10.5772/50527

© 2012 Montiel-Ross et al.; licensee InTech. This is an open access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/3.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Abstract In this paper, an efficient methodology to generate optimal and/or quasi-optimal sequences of G commands to minimize the manufacturing time is presented. Our solution starts from original G codes provided by application CAD/CAM software. Here, first we tackled the problem of reducing the time of the travel path for drilling of an industrial robotic manufacturing machine. The methodology can be easily implemented for free distribution or commercial CAD/CAM software without achieving any modification to it. Several experiments that demonstrate how this proposal can help to outperform solutions provided by application software are presented, consistent improvements around 62% were obtained. Moreover, for optimizing the time along the travel path, we present a high performance implementation of Ant Colonies (ACO) known as Parallel ACO (P-ACO) that allows achieving the optimization task efficiently by speeding up the original ACO. A Graphical User Interface that integrates the whole process is shown.

Keywords Manufacturing robot, P-ACO, G-CODE

1. 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 [1], reducing time and avoiding human errors. They are, in essence, manufacturing robots.

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, one of those processes being the hole cutting operation or drilling.
Several studies published since 1996 are based on reducing the cutting time by optimizing some parameters, such as geometry, material and tool type. Some works on drilling route optimization investigated: [2] a case study with variable hole sizes, in [3] using Tabu search algorithm and an improved algorithm; in [4] the 2-Opt Heuristic Evolutionary algorithm in optimizing the drilling path; in [5] a Genetic Algorithm (GA) to find the optimal sequence for the hole cutting process of the CNC machine tool and in [6] a Particle Swarm Optimization (PSO) approach to Robotic Drill Route Optimization. A survey of the literature shows that much research has been done on minimizing the cutting time [7,8]. In [9] the travel path is formulated as a special case of the travelling salesman problem.

Notwithstanding the importance for industry of reducing the travel path time, there have been few new proposals that face this problem efficiently. In [10] a novel scheme based on an adaptive grid that can produce the optimal tool path based on an iso-parametric tool path was introduced. In [11] a new method for rapid tool movement in CNC machines by using image processing techniques was presented.

With the advent of novel architectures regarding high performance computing [12], several studies about parallel metaheuristics have been developed, e.g., in [13] a synchronous fine grained parallel Ant Colony Optimization (ACO) algorithm in master/slave fashion was presented, combined with local Tabu search, and this algorithm was applied to solve the Quadratic Assignment Problem [14]. In [15] five parallelization strategies and the implementation of one of them, named parallel ants, were shown. By modifying the classical ACO [16], in [17] a parallel ant colony algorithm on reconfigurable processor arrays was proposed. A multicore processors methodological design for CAD applications proposed to solve the min-cost flow problem was described in [18]. An approach of tool path generation for CNC machining based on CUDA was explained in [19]. New scientific and technological trends show the interest in using bio-inspired algorithms to solve real-world problems, for example, in [20] ACO combined with a Fuzzy Logic controller tuned with the Simple Tuning algorithm [21,22] was used to obtain the optimal route for a mobile robot, in [23] also the parameters of a fuzzy controller were tuned, and ACO was used for a mobile robot, in [24,25] the parameters of fuzzy controllers were optimized using ACO. In [26] the implementation of a methodology that combines Fuzzy Logic and Genetic Algorithms to plan the automatic assembly and disassembly sequence of products was presented.

In this paper, we present an efficient methodology to generate the best sequence of G commands for a set of holes of a printed circuit board, which is important since knowing the optimal sequence makes it possible to reduce production costs. This methodology consists of reading a DXF file to obtain a set of commands and coordinates of any part (section) designed with an application CAD/CAM package; this input file is processed by the Parallel ACO (P-ACO) algorithm to generate an optimal sequence of G commands for a CNC drilling tool path, then the machine is fed with the optimal sequence and the manufacturing task is performed. The path optimization is achieved using Multi Ant Parallel Colonies Metaheuristic which is an efficient parallel implementation of ACO [27]. Several experiments that demonstrate how this proposal can help to outperform solutions provided by application paid software are presented. We propose a strategy for information exchange randomly between processors, updating pheromone values automatically.

This paper is organized in seven sections. Section I discusses the state of the art drilling route optimization problem. Section II presents the route optimization problem focusing on how to optimize the travel time path of the drilling tool; it is viewed as an analogous problem of the Travel Salesman Problem (TSP). In Section III, the mathematical foundation of ACO and its pseudo code are presented. Section IV extends ACO theory to P-ACO, its pseudo code, and the Graphical User Interface (GUI) of this development is also presented. Section V explains the proposed methodology providing the main steps and algorithms that allow duplicating results. In Section VI, we explain how the experiments were conducted and numerical and graphical information that support the results is shown. Finally, in Section VII we give the conclusions of this work.

2. Problem Formulation

The TSP problem [14,28,29] is the problem of a salesman who, starting from his hometown, wants to find a shortest route that takes him through a given set of customer cities and then back home, visiting each customer city exactly once. The TSP can be represented by a complete weighted graph $G = (V; E)$, with $V$ being the set of vertices (nodes) representing the cities and $E$ being the set of arcs, that can be implemented using an adjacency matrix representing the connections between nodes. Each arc $(i, j) \in E$ has assigned a value (length) $d_{i,j}$ which is the distance between cities $i$ and $j$.

Route optimization can be achieved using metaheuristics; ACO is one of the many available algorithms used to optimize routing time; numerous studies on hole drilling optimization do not consider the treatment of the problem using Parallel Ant Colony Optimization (P-ACO) [15].
Ant Colony Optimization (ACO) was introduced by Marco Dorigo [29-31,34]. Using very simple communication mechanisms, an ant group can find the shortest path between any two points by choosing the paths according to pheromone levels. ACO metaheuristics can be used to solve the TSP, where the pheromone trails are associated with arcs and therefore $\tau_{ij}$ refers to the desirability of visiting city $j$ directly after city $i$. For Ant Systems (AS), 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 $\tau_{ij}$'s.

Routes 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 value to probabilistically construct a tour by iteratively adding cities that the ant has not yet visited, until all cities have been visited.
3) Each ant goes back to the initial city.
4) After all ants have completed their route, they may deposit pheromones on the routes they have followed.

ACO can usually converge to an almost optimal solution in a certain number of iterations, but it needs considerable computational time and resources when the complexity of the problem increases [32, 33].

3. Ant System

Consider the general problem of finding the shortest path between two nodes on a graph, $G = (V, E)$ with $n_G = |V|$ nodes. The length, $L^k$, of the path constructed by ant $k$ is calculated as the number of hops in the path from the origin to the destination node, as is shown in Figure 1, where each edge $(i,j)$ of the graph has a total associated pheromone concentration, $\tau_{ij}$.

The main difference between AS and Simple Ant Colony Optimization (S-ACO) [34] is based on the transition probability $P^k_{ij}$. AS includes heuristic information and implements a memory capability by the inclusion of a Tabu list.

A small random initial value of pheromone $\tau_{ij}(0)$ and heuristic information $\eta_{ij}$ are assigned to each edge; this is indispensable to use (1) because at the first step they do not have any pheromone concentration value. An ant randomly selects which edge to follow next. Using the simple decision logic based on (1).

A number of ants, $k = 1, \ldots, n_k$, are placed on the source node. For each iteration of AS, each ant incrementally constructs a path (solution) to the destination node. At each node, each ant executes a decision policy to determine the next link of the path; if ant $k$ is currently located at node $i$, it selects the next node $j \in N^k_i$ based on the transition probability [30]. In AS, the probability of moving from node $i$ to $j$ is given as

$$P^k_{ij} = \frac{[\tau_{ij}]^\alpha[\eta_{ij}]^\beta}{\sum_{l \in N^k_i}[\tau_{il}]^\alpha[\eta_{il}]^\beta}, \quad \text{if } j \in N^k_i$$  \quad (1)

where $\eta_{ik} = 1/d_{ij}$ is a heuristic value, $\alpha$ and $\beta$ are two parameters, which determine the relative influence of the pheromone trail and the heuristic information, and $N^k_i$ is the feasible neighbourhood of ant $k$ when being at city $i$, that is, the set of cities that ant $k$ has not yet visited. The probability of choosing a city outside $N^k_i$ is 0. The role of parameters $\alpha$ and $\beta$ can be set by the following statements:

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

The transition probability in equation (1) differs from that of S-ACO in two aspects [34]:

- The transition probability used by AS is a balance between pheromone intensity, $\tau_{ij}$, and heuristic information (desirability of the move) $\eta_{ij}$. This effectively balances the exploration - exploitation trade-off. The heuristic information adds an explicit bias towards the most attractive solutions and it is, therefore, a problem-dependent function.
- The set $N^k_i$ defines the set of feasible nodes for ant $k$ when located on node $i$. To prevent loops, $N^k_i$ may include all nodes not yet visited by ant $k$. As the ant visits a new node, that node is
added to the ant’s Tabu list. Nodes in the Tabu list are removed from \( N^k_i \), ensuring that no node is visited more than once.

### 3.1 Update of pheromone trails

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

Pheromone evaporation is implemented by

\[
\tau_{ij} \leftarrow (1 - \rho)\tau_{ij}, \quad \forall (i, j) \in L,
\]

where \( 0 \leq \rho \leq 1 \) is the pheromone evaporation rate [30]. After evaporation, all ants deposit pheromone on the arcs they have crossed, as shown in (3)

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

where \( \Delta \tau_{ij}^k \) is the amount of pheromone that ant \( k \) deposits on the visited arcs. Algorithm 1 describes the general sequential procedure of the Ant System.

### 4. Parallel ACO Framework

#### 4.1 Parallelism on the Ant Level

In this research, we use a parallel implementation of ACO (P-ACO) on the ant level [34] proposed initially by Bolondi and Bondanza [35], as shown in Figure 2. A master–slave architecture is used to implement ant level parallelization.

![Figure 2. Ant Level Parallelization by Bolondi and Bondanza.](image)

Each ant (slave) is assigned to a separate processor to construct a solution. One processor is assigned as the master, with the responsibility to schedule ants on the different processors, to place ants on the representation graph (in this case, to place ants randomly), to perform global updates, to produce the final solution and to manage user input.

Each slave is responsible for constructing a complete path (solution) and for performing a local search on that solution if applicable. The solution is then sent to the master. Additionally, since each slave maintains its own pheromone structure and updates it, the changed pheromone values are sent to all other processors at the end of each iteration, as shown in Figure 2.

Algorithm 2 represents our P-ACO Framework based on 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 route found so far. A new iteration is initiated until stopping criterion is met.

Ignoring any communication overhead, parallel algorithms would imply optimum (asymptotic) speedup [36]; assuming that an infinite number of processing elements is available. The main part of the algorithm is complexity \( O(m^3) \) and the generation of one solution is complexity \( O(m^2) \) where \( m \) is the number of jobs; these two operations are independent of each ant of a given cycle, so they can be easily parallelized. Asymptotic speedup is based on the relation between sequential and parallel execution times (4), where \( m \) is the quantity of ants in the colony, \( T_{seq} \) is the sequential time and \( T_{par} \) is the parallel execution time.

\[
S_{asymptotic}(m) = \frac{T_{seq}(m)}{T_{par}(m, \infty)} = \frac{O(m^3)}{O(m^2)} = O(m)
\]
Algorithm 2 Parallel ACO for TSP

Procedure P-ACO(m, n, S, p)
  
m : number of ants
  n : number of cities
  S : feasible nodes (cities) for ant k
  p : number of processors
  Initialize all parameters, i.e., α, β, ρ, τ
  repeat
    for each processor do
      for each ant k := (1,..., m) solution construction do
        S := 1,..., n set of selectable cities
        Choose city i with probability \( p_{0i} \)
        repeat
          Choose city j ∈ S with probability \( p_{ij} \)
        end for
        S := S − j
        i := j
        until \( S = 0 \)
      for all i,j in iteration best solution do
        \( \tau_{ij} := \tau_{ij} + \Delta \) (intensification)
      end for
    until Stopping criterion is met

Algorithm 2. Parallel ACO for TSP.

In parallel computing, running time depends on the number of processors used, and derived measures are often used to express whether a particular execution time is good or bad considering the amount of resources used. Speedup is used to express how many times our parallel algorithm works faster than a sequential one, where both algorithms (sequential and parallel) are solving the same problem [37].

The parallel algorithm (P-ACO) is written in Visual C#, avoiding some synchronization problems related to parallel programming. The main application software works on single processor machines and dual-processor machines; these experiments have been executed with the following hardware specifications:

- Intel® Core™ i7 CPU 930 @2.80 GHz processor
- 6GB RAM memory

5. CNC Tool Path Generation

The TSP and Parallel ACO have been incorporated to find the shortest cutting tool travel path (CTTP) for drilling operations. Figure 3 shows our Graphics User Interface (GUI) developed with Visual C#. A number of commercial CAD/CAM 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 CTTP should be determined [9].

The total production time to cut any part using CNC machine tools consists of a travel time [9,19], which is the time to move the CNC machine spindle between operations, switch time, which is the time to change the tool for the next operation, and the cutting time, which is the time that the tool moves with cutting speed in the air or on material. Our purpose is to minimize the CTTP based on P-ACO.

![Figure 3. CNC Tool path optimization GUI - note that each hole is treated as if it were a city in the Travel Salesman Problem (TSP).](image)

5.1 Tool path programming optimization strategy

To exchange files between CAD/CAM packages, a translator is necessary [38,39] which takes place through intermediate files, as shown in Figure 4. These files can be any of the following standard formats: Drawing Exchange Files (DXF), IGES files and STEP files.

![Figure 4. Exchange files between CAD/CAM packages.](image)

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 Figure 5.

![Figure 5. Automatic G code sequence generation from DXF files using P-ACO.](image)

5.2 DXF file interpretation

For the input DXF file shown in Figure 6, a translation procedure must be done. This procedure can be represented as in Algorithm 3.
The ReadFromFile(string filename) method deals with extracting data from the DXF file and forwarding data to the necessary drawing methods. In DXF files, the data is stored in two-line structures, as shown in Figure 7. The first line contains an entity identifier and the second line contains the data value.

5.3 Automatic G - Code Generation for the CTTP

The following procedure represents the generation of a G code sequence for the CTTP:

- **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 moves 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 tool reaches the last hole in the CTTP.

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

6. Experimental Results and Analysis

6.1 Parallel Implementation of ACO Analysis

In this section, we show the test results of our parallel ACO algorithm on some \( 15 \times 15 \) boards with a different number of holes. We ran our parallel algorithm for 97 holes, 121 holes and 200 holes with the following decision probability control parameters:

- \( \rho = 0.1 \) Evaporation coefficient.
- \( \alpha = 1, 2, 5 \) Parameter to regulate the influence of \( \tau_{ij} \).
- \( \beta = 4, 4.5 \) Parameter to regulate the influence of \( \eta_{ij} \).

The number of ants was equal to the number of iterations and the maximum number of processors was 6. In our experiment we performed 15 trials on each problem. The experimental results are shown in Figures 8, 9 and 10, where three problems were tested based on P-ACO, and they are compared with a classical ACO algorithm by using a single processor.
Figures 8, 9 and 10 show the results of our experiments; the number of ants varies from 30 to 50 and the number of iterations is equal to the number of ants. For each processor configuration, the time indicated is the average of 15 independent runs.

The number of processors plays a major role in determining the complexity of a parallel algorithm. In general, we say that the performance of a parallel algorithm is expressed in terms of how fast it is and how many resources it uses when it runs. The sequential and parallel execution times for these three experiments slowly increased as the system size increased, but their complexity is reduced in the parallel algorithms; the execution times for all experiments in Figures 8, 9 and 10 follow the same trends; the execution time consistently decreases as the number of processors used increases.

It can easily be seen from Figures 8, 9 and 10 that P-ACO computation is reduced due to the parallel computation. P-ACO implementation provides good solutions and they are near cost-optimal and scalable. We can observe that when the number of processors is increased, the computing time can be reduced due to the small number of ants assigned to each processor; due to the overhead of communication which increases the total time of an algorithm, the speedup of our algorithm shows a good trend as indicated in Figure 11.

6.2 Tool Path Optimization Analysis

In this section, we are going to use the same three experiments in section A, with the aim of comparing the generated tool paths using the P-ACO algorithm against the MasterCAM™, the comparisons are shown in Figures 12, 13 and 14. We can observe that the determination of an optimal tool path is a good example of the well-known TSP problem and it can be solved by using a metaheuristic; in this case, we use P-ACO to perform these experiments; here the machining head must meet an optimal set of machining operations.

Figure 15 shows a plot of cutting time vs. number of holes; we can observe that for a 97 hole board, there is a difference of 24.6 minutes to travel a tool path between P-ACO and MasterCAM™; for a 121 hole board there is a difference of 27.35 minutes and for a 200 hole board there is a difference of 51.88 minutes. Figure 16 shows a HAAS Automation™ CNC machine used to perform our experiments.
Table 1 shows an estimated travel time comparison between MasterCAM™ and P-ACO to perform drilling operations. We can observe that using P-ACO the CNC machine performs better, i.e., 62.27% faster than some paths generated by MasterCAM™. We present an estimate of machine travel times for more boards based on the results obtained with a single board, as shown in Figure 15.

7. Conclusion

In this paper, we presented an efficient methodology to improve application software, which can be free or paid software for industry applications. It is based on finding optimal time paths for CNC machines providing optimal G code. The solution takes advantage of high performance computing for implementing the P-ACO metaheuristics. Several experiments were achieved using paid application CAD/CAM software, the results were verified using a professional CNC machine (see Figure 16) and the analysis of numerical results and graphical trends demonstrates utterly that the use of the proposed methodology can offer improvements of around 62% over solutions provided by application software, in this case paid software. This methodology can be easily implemented, as there is no necessity to perform any modification to the application software, since the only thing to do is to feed the CNC machine with the optimal G code sequence instead of the original.

8. References

[1] Mattson M. (2010) CNC Programming: Principles and Applications. Delmar Cengage Learning, USA.
[2] Kolahan F, Liang M (1996) Tabu search approach to optimization of drilling operations, Computers and Industrial Engineering. Vol. 1-2, No. 31, pp. 371 - 374.
[3] Kolahan F, Liang M (2000) Optimization of hole-making operations: a tabu search approach, International Journal of Machine Tools and Manufacture, Vol. 2, No. 40, pp. 1735-1753.
[4] Sigl S, Mayer H. A (2005) Hybrid Evolutionary Approaches to CNC Drill Route Optimization, Proceedings of Computational Intelligence for Modelling, Control and Automation, pp. 905 – 910.
[5] Qudeiri J. A, Yamamoto H, Ramli R (2007) Optimization of operation sequence in CNC machine tools using genetic algorithm, Journal of Advanced Mechanical Design, Systems and Manufacturing, Vol. 1, No. 2, pp. 272-282.
[6] Asrul A, Amar F. Z. A, Zuwairie I, Abdul R. H, Zulkifli M. Y, Ismail I (2010) A particle swarm optimization approach to robotic drill route optimization, Fourth Asia International Conference on Mathematical, Analytical Modelling and Computer Simulation, pp. 60-64.
[7] Castelino K, Wright P. K (2003) Tool path optimization for minimizing airtime during machining, Journal of Manufacturing Systems, Vol. 22, No. 3, pp. 173-180.
[8] Kolohan F, M. Liang (2000) Optimization of hole-making operations: a Tabu-search approach, International Journal of Machine Tools and Manufacture, Vol.40, No. 12, pp. 1735-1753.

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

[10] Wei He, Ming Lei, Jongzhan Bin. (2009) Iso-parametric CNC tool path optimization based on adaptive grid generation. The International Journal of Advanced Manufacturing Technology, 45:538-548.

[11] Shahed Shojaeipour (2010) A Novel Method for Automated Tool Path Optimization for CNC Machining Operations. Solid State Phenomena 357. pp. 166-167.

[12] Munir, A., Ranka, S., Gordon-Ross, A (2012) High-Performance Energy-Efficient Multicore Embedded Computing. IEEE Transactions on Parallel and Distributed Systems. 23(4) pp. 684-700.

[13] Talbi, E.G. et al. (2001) Parallel Ant Colonies for the quadratic assignment Problem. Future Generation Computer Systems. 17(4): pp. 441-449.

[14] Randall M, Montgomery J (2002) Candidate Set Strategies for Ant Colony Optimization. Technical Report, School of Information Technology, Bond University.

[15] Randall M, Lewis A (2004) A Parallel Implementation of Ant Colony Optimization, Journal of Parallel and Distributed Computing, 62: pp. 1421 - 1432.

[16] Dorigo M, Caro G. Di (1999) The Ant Colony Optimization Meta-Heuristic. In D. Corne, M. Dorigo, and F. Glover, editors, New ideas in Optimization. McGraw-Hill, pp: 11 - 32.

[17] Merkle D, Middendorf M (2002) Fast Ant Colony Optimization on Runtime Reconfigurable Processor Arrays, Genetic Programming and Evolvable Machines. 3(4): pp. 345 - 361.

[18] Yinghai Lu, Hai Zhou, Li Shang, Xuan Zeng (2010) Multicore Parallelization of Min-Cost Flow for CAD Applications. IEEE Trans. on CAD of Integrated Circuits and Systems. 29(10): 1546-1557.

[19] Yu Wujia, Zhou Yajun (2010) An Approach of Tool Paths Generation for CNC Machining Based on CUDA. 2nd International Conference on Information Science and Engineering (ICISE), pp. 4925-4928.

[20] Miguel Angel Porta Garcia, Oscar Montiel, Oscar Castillo, Roberto Sepúlveda (2007) Optimal Path Planning for Autonomous Mobile Robot Navigation Using Ant Colony Optimization and a Fuzzy Cost Function Evaluation. Analysis and Design of Intelligent Systems using Soft Computing Techniques 2007: pp. 790-798.

[21] Miguel Angel Porta Garcia, Iliana Marlen Meza Sanchez, Oscar Montiel, Roberto Sepúlveda, Oscar Castillo (2006) Simple Tuning of a Fuzzy Pulse Width Modulation Controller for a DC Motor Application. in Proceedings of the 2006 International Conference on Artificial. Intelligence, ICAI 2006, Las Vegas, Nevada, USA, June 26-29, 2006, Vol. 2, pp. 598-604.

[22] Ortiz-de-la-Vega H.A, Gomez-Ramirez E, Cortes-Rios J.C (2010) Simple Tuning Algorithm improvements for fuzzy logic controllers, 2010 IEEE International Conference on Fuzzy Systems. pp.1-8, 18-23 July 2010.

[23] Castillo Oscar, Martinez-Marroquin Ricardo, Soria José (2009) Parameter Tuning of Membership Functions of a Fuzzy Logic Controller for an Autonomous Wheeled Mobile Robot Using Ant Colony Optimization. SMC 2009: pp. 4770-4775.

[24] Castillo Oscar, Martinez-Marroquin Ricardo, Melin Patricia, Valdez Fervier, Soria José (2012) Comparative study of bio-inspired algorithms applied to the optimization of type-1 and type-2 fuzzy controllers for an autonomous mobile robot. Inf. Sci. 192: 19-38.

[25] Naredo Enrique, Castillo Oscar (2011) ACO-Tuning of a Fuzzy Controller for the Ball and Beam Problem. MICAI (2) 2011: 58-69.

[26] Galantucci L.M, Percoco G, Spina R (2008) Assembly and Disassembly Planning by using Fuzzy Logic & Genetic Algorithms, International Journal of Advanced Robotic Systems, Vedran Kordic, Aleksandar Lazinica and Munir Merdan (Ed.), ISSN: 1729-8806, IntTech, Available from: http://www.intechopen.com/journals/international_journal_of_advanced_robotic_systems/assembly_and_disassembly_planning_by_using_fuzzy_logic_amp_genetic_algorithms

[27] Stützle T (1998) Parallelization Strategies for Ant Colony Optimization. In: A.E. Eiben, T. Böck, M. Schoenauer, and H-P. Schwefel, editors, Proceedings of the Parallel Problem Solving from Nature Conference, In: Lecture Notes in Computer Science, volume 1498, pp. 722 - 731. Springer - Verlag, Germany.

[28] Dorigo M, Gambardella L.M (1997) Ant Colony System: a cooperative learning approach to the traveling salesman problem. IEEE Trans. On Evolutionary Computation. 28:53-66.

[29] Dorigo M, L.M. Gambardella (1997). Ant Colonies for the Traveling Salesman Problem. BioSystems, Université Libre de Bruxelles, Belgium. 43, 73-81.

[30] Dorigo M, Stützle T. (2004) Ant Colony Optimization. MIT Press, Cambridge, MA.

[31] Dorigo M, Stützle T (1996) An experimental Study of the Simple Ant Colony Optimization of Cooperating Agents. IEEE Transactions on Systems, Man, and Cybernetics - Part B, 26(1): 29-41.
[32] Jie Xiong, Caiyun Liu, Zhong Chen (2008) A New Parallel Ant Colony Optimization Algorithm Based on Message Passing Interface. PACIIA (2), pp. 178-182.

[33] Cecilia J.M, García J.M, Ujaldon M, Nisbet A, Amos M (2011): Parallelization Strategies for Ant Colony Optimization on GPUs. IPDPS Workshops 2011, pp. 339-346.

[34] Engelbrecht A.P (2006) Fundamentals of Computational Swarm Intelligence, University of Pretoria, South Africa, Wiley & Sons.

[35] Bolondi M, Bondanza M (1993) Paralleizzazione di un Algoritmo per la Risoluzione del Problema del Commesso Viaggiatore. Master's thesis, Politecnico di Milano, Italy.

[36] Bullnheimer B, Kotsis G, Strauss C (1998) Parallelization Strategies for the Ant System. High Performance Algorithms and Software. In: Nonlinear Optimization, R. De Leone, A. Murli, P. Pardalos, and G. Toraldo, editors. Series: Applied Optimization, Kluwer: Dordrecht, Holland. vol. 24, pp. 87-100.

[37] Padua David (2011) Encyclopedia of Parallel Computing. Urbana, Illinois, USA. Springer.

[38] Radhakrishnan P., S. Subramanyan, V. Raju (2000) CAD/CAM/CIM. New Age International Publishers, 2nd ed, New Delhi, India.

[39] Zude Zhou, Shane S. X, Dejun Chen (2011) Computing Manufacturing in Digital Manufacturing Science Digital, Springer - Series in Advance Manufacturing, Verlag, London, pp: 57-104.