Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 5, Issue. 2, February 2016, pg.7 – 15

Document Image Segmentation using Multi ANT Colonies Algorithm (MAC) on a Multi-Core Processor

[1]Hasanen S. Abdullah, [2]Ammar H. Jasim
[1]Computer sciences Department, University of Technology, Iraq
[2]Department of computer Science, University of Baghdad, Iraq
[1]qhasanen@yahoo.com, [2]ammar_hussein_2004@yahoo.com

***Abstract*—this paper develops a special design for parallelization strategy of Ant Colony Optimization (ACO) algorithm for document image segmentation on a machine with a multi Core processor for fast processing to find solutions. This work focus on the advantage of parallelism property in ACO algorithm. Therefore, we will use multiple colonies using Java thread programming technique. According to the synchronization and exclusive control modes among threads, indirect communication between ants on different colony (asynchronous parallel) is proposed.**

This work proposes a MAC algorithm for searching process that based on sequential version of ACO algorithm. The proposed algorithm composed of several independent colonies work together (simultaneously) for document image segmentation utilizing the benefit of multi Core CPU, while the standard algorithm has single ant colony. The design algorithm shows good results than the sequential original algorithm.

In this paper we will design a parallelization approach of a modified ACO algorithm for document image segmentation. This design aim to finding the optimal segmentation with a low computation time. The mapped image is segmented by colonies independently; finally the solutions of each colony are combining to obtain the full segmentation. According to the overhead impose by a synchronization of individuals (ant) and colonies, we design parallel models for platform with a multi-core processor utilize the Hyper-Threading technique which motivates our design for parallelization.

Keywords*— *ACO (Ant Colony Optimization), Ant Colony System (ACS), Multiple Ant Colony (MAC), Multi ACO, Max–Min Ant System (MMAS), Multi Core, Graphics Processing Units (GPU), Feature vector, Document Image Segmentation, Image segmentation.

I. INTRODUCTION

Multiple ACO strategy considered as a new direction of ACO algorithm researches that designed to increase the performance of Ant colony algorithms. In this strategy multiple colonies with its ants are cooperated to solve a combinatorial optimization problem [1, 2]. MACO increases the chance of ant colony algorithms for searching large search space area and hopefully discover optimal solution.

The Ant colony algorithms parallel architecture can be done through the increase of velocity of construction stage, the search space decomposition or problem domain, or create distributed colony system (multiple colony methodology). D. and D. Caro [3] In addition to D. and S. [4] provide an outline of several parallel approaches implementations. Most models of parallelization can be categorized into fine grained models and coarse grained models. In the fine-grained model, the population of ants is separated into a large number of very small sub-populations. These are maintained by different processors, and some information is exchanged frequently among those processors. This model is suitable for massively parallel architecture-systems with a huge number of processors connected with a specific high speed topology. In the coarse-grained model, the population of ants is divided into few sub-populations. These sub-populations are maintained by different processors and some information is exchanged among those processors at every defined exchange step. This model is suitable for clustered computers.

The common ACO algorithm consists of three phases (see Fig. 1). After the initialization and until some termination condition is met the following steps are repeated: each ant builds solutions, the best(s) solution(s) are improved by a local search (this step is optional) and at last the pheromone trail is updated.

```
initialize the parameters
    initialize the pheromone trail
    while termination criteria not met do
    construct ant solutions
    apply local search (optional)
   global pheromone trail update
end_while
```

Fig. 1 The ACO Metaheuristic

Parallel processing considers a perfect approach to solve effectively considerable optimization problems [5, 6]. The structure of sequential ACO algorithms is highly suitable for parallelization. During each iteration, the single ant behavior is entirely independent of all other ants' behavior through that iteration. The independent behavior of ants exploited to build a strategy for parallelization. The main parallel ways of applying the ant colony algorithm are achieved by using multi-core CPUs [7, 8]. However, the programming models of these ways are complex and their excellent performances are limited by the number of CPU cores.

Currently new hardware types have been useful for high implementation. Among them, the multicore CPU and GPU provide great performance computing power with inexpensive and reasonable cost while programming this types of hardware are difficult. In fact, the conventional models are not appropriate for using parallelism in such a way that become efficiently applied on these type of hardware.

As a test problem we use our proposed document image segmentation algorithm and compare the behavior of multiple independent ant colonies with single ant colony algorithm.

II. RELATED WORK

Significant solutions are presented to enhance the ACO algorithm performance. Among those suggested solutions, we discover the utilization for parallel computing to decrease calculation time and improve the quality of solution or both. We will address to mention some of these researches:

Most implementations of parallel ant colony algorithms can be categorized into two approaches. First one is the parallel implementations of construction phase using single colony. Launched by Bullnheimer [9], it intent to increase computations by distributes the ants to processing elements. The second approach, presented by Stützle [10], using MAC. In this situation, each ant colonies are allocated to processors to increase the computations speed also to improve quality of solution by using cooperation mechanism between colonies.

Stutzle [10] described the simplest case of parallelization, parallel independent ACO searches that do not interact; this is the largest scale of parallelization where entire searches can be performed concurrently. Michel and Middendorf [11] discuss exchanging information, where separate ant colonies exchange trail information pheromone matrices.

Other general parallelization strategies include different scaling levels, where parallelism can be exploited at different scales. Parallel independent ant colonies, as described in [10], have no communication overhead at all.

Pedemonte [12] proposed a detailed classification of parallel ant colony approaches. It shows the most works dependent on the parallel design of ant colony algorithm that suitable for conventional parallel machines.

Scheuermann [13, 14] they used Field Programmable Gate Arrays (FPGA) to design and implement the parallel algorithm for ant colony. They show the benefit of using this type of architecture to implements parallel ant colony.

Catala et al. [15] propose an implementation of ACO to solve the Orienteering Problem. Instances of up to a few thousand nodes are solved by building solutions on GPU. Wang et al. [16] propose an implementation of the MMAS where the tour construction phase is executed on a GPU to solve a 30 city Travelling Salesman Problem (TSP). Following these works, Delévacq et al. [17] proposed different ACO parallelization strategies on GPU and used a comparative study to demonstrate the search efficiency is influence by several parameters.

Randall et al.[18] proposed an interesting classification of parallelization strategies for ACO metaheuristic. These are: Parallel Independent Ant Colonies, Parallel Interacting Ant Colonies, Parallel Ants, and Parallel Evaluation of Solution Elements.

Bullnheimer[19] present two parallel executions of AS algorithm, the first one called Synchronous Parallelism Implementation use master–slave paradigm in which each ant work as a slave to discover the solution and then transmit the result to a master.

The second called Partially Asynchronous Parallelism Implementation.

Stutzle [10] shows the efficiency of utilizing parallel independent against sequential implementations of the MMAS algorithm to solve the TSP. Middendorf [20,21] presented a parallel implementation, using coarse-grained paradigm, of the Ant System. Their multiple ant colony algorithm utilizes best-exchange design in which the best solutions is exchanged between ant colonies in two different mechanism. Chu [22] applies a parallel implementation, using coarse-grained paradigm, of ACS and utilized the same suggestion of exchange best solution but under different communication topologies. Rahoual [23] shows two techniques for parallel Ant System algorithms with local search to solve the problem. The first technique applies the algorithm using separated processors units, which work independently. The second technique, every ant was considered as a process and is allocated for one processor. Talbi [24] show a different master–slave model to parallelize Ant colony algorithm in which each slave use a local search technique depending on the "tabu" search to enhance the solution found by each ant.

III. THE PROPOSED ALGORITHM

In this section the design steps of our system are illustrated, that divided into two sections: First one describes the proposed document image segmentation using single ant colony algorithm, and the second will demonstrate the design of proposed multi ant colonies algorithm.

A. Proposed Method for Document Image Segmentation

For document image segmentation we will focus on the characteristics feature of text, image, and background region. These characteristics referred to as features, are extracted to identify text, image, and background objects. our observation include simple and uncomplicated statistical features, that are, mean, standard deviation and intensity of block pixel gray scale to discriminate those objects from one another. These statistics were utilized as the feature values of pixel colors. Pixel will converted to a corresponding gray scale level (tuple of three values mean, standard deviation and intensity).

Using the following observations, we can segment the pixel of document to text, image, and background pixel as follows:

1. Pixels representing a textual area have higher feature values than their background and high standard deviation.
2. Pixels representing a picture/image area have low mean and standard deviation since the gray scale level usually appear darker than those of textual pixels.
3. Pixels representing a background area close to zero standard deviation and lighter background gray scale level.

The flowchart in Fig. 2 summarizes the main step in proposed algorithm based on ACO algorithm.

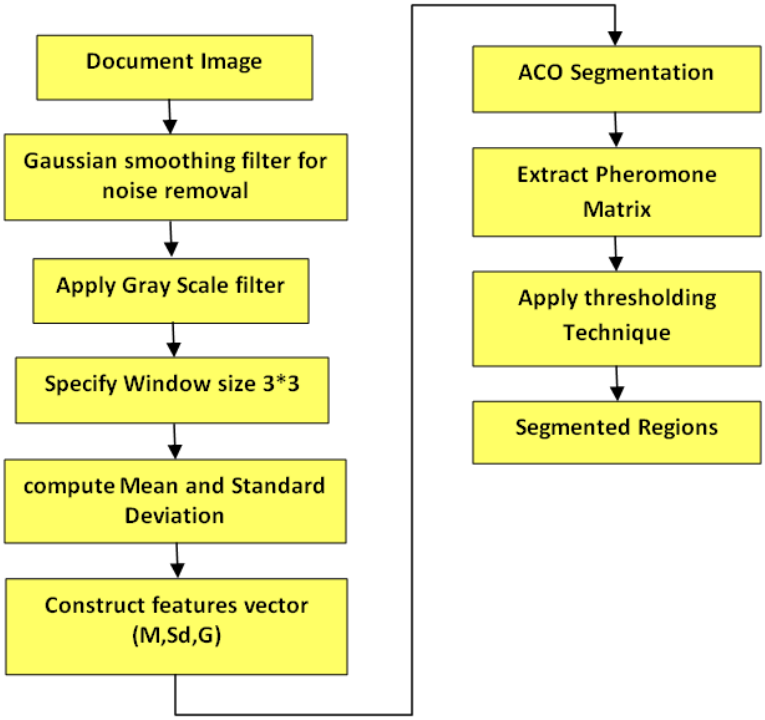

Fig. 2 Main Steps for Ant Segmentation System

B. Proposed MACO Algorithm for Document Image Segmentation

In this Section we describe the designs for the multi Ant System, as applied to the document image segmentation. The design approach is an asynchronous master/slave concept based on multi core architecture. One core is used as master core and other cores as slaves. The master Core contains control thread which responsible for initializes the environment for colonies, broadcast the initial parameter and pheromone matrix and receive the best found solution (the pheromone matrix) from colonies. The slave cores contain colonies that each of them contain a number of independent ants implements the search process. The parallel algorithm works as follows: The initial pheromone matrix and parameter will be managed by the master and broadcasts to all the colonies. At each iteration, each ant in each colony constructs the solution. Each slave receives the pheromone matrix, constructs a complete solution, and sends the final solution found in each colony to the master. When the master receives all the solutions, combines the pheromone matrixes and then displays it.

In this algorithm, the searching process for each ant treat as independent process, since there is no information exchange or interference between ants; we can simply parallelize this phase. However, main thread needs the result from each colony. Therefore, master thread wait until the entire ant's thread of ACO is computed and then the result is passed to the master thread.

The proposed approach exploits a number of colony each of them contain K ants, which moves on the image based on local difference of statistics features of neighboring pixels.

C. Proposed Multi Ant Colony Algorithm

This algorithm is a multiple colony inspired in the Ant colony system. The most appropriate features of our proposal are:

1. Proposed algorithm use several colonies
   (a) Each colony has the same number of ants and iterations
   (b) Each colony uses the same initial pheromone matrix.
2. Each colony has the same set of parameters (α, β, ρ, and φ).
3. Each colony work independently from other.

The proposed algorithm is show in Fig. 3.


```
Master ACO algorithm
parameter_initialization
Pheromone matrix initialize to τ0
Mapped document image to grayscale level where
Each pixel represent as a tuple (pixel (μ,σ ,I))
Multicast the initialized information and grayscale level document image to all colony type (text, image and
background colony)
For each C=1...3 do   // C represent the number of colony according to
                         this classification (text, image and background)
    Distribute randomly K ant on image
     While termination criteria not satisfy (iteration_max ) do
          For each ant k=1...m do
             Create new_thread_ant(k)
            Check Colony_type
            If (background_colony)
                ant  construct a background_solutions
            end_if
            if (Image_colony)
               ant  construct a Image_solutions
            end_if
            if (Text_colony)
               ant  construct a Text_solutions
            end_if
            local update pheromone (update rule)
          end_for
          global update pheromone trails (global update rule)
      end_while
Send best solution
end_for
While no solution arrive from all colony
Wait for a solution;
End while
```

Fig. 3 The Multi Ant Colony algorithm

Our conception contains the following mechanisms, when each ant in different colonies found its corresponding pixel, label current pixel with its specific segment type without sending information about labeled pixels to other colonies to overcome the communication overhead and in the final step, collect pheromone matrices values and put them in one matrix to be displayed.
The slave ACO for document segmentation as shown in Fig 4.

```
Received pheromone values
Received image
Repeat
   For ant k ∈ {1 ...M} {solution construction}
      S = {1, ... ... ... .8} {Set of neighbor pixels}
      Choose pixel i with probability P_i
      Repeat
        Choose pixel j ∈ S with max probability P_ij
        Deposit a specific amount of pheromone according to type of colony
   Endfor
   Forall i, j do
      τ_i,j = (1 − ρ). τ_i,j {Vanishing}
   Endfor
   Forall i; j in iteration best solution do
        τ_i,j = τ_i,j + η_i,j. Δτ_i,j {Increase}
   Endfor
Until stopping criterion is met (max_iteration)
Send to master the best solution found;
End
```

Fig. 4 The Slave ACO Algorithm

Heuristic measure is the key process used in the construction process It is proposed to be determined by the local statistics features at the pixel position (i, j) and given by the equation (1):

$$\eta_{i,j} = \frac{1}{d_{i,j} \cdot \Delta I_{i,j}} \dots\dots\dots\dots\dots\dots (1)$$

$$d_{ij} = \sqrt{(i-j)^T . C^{-1}(i-j)} \dots\dots\dots\dots\dots\dots (2)$$

$\boldsymbol{\eta_{i,j}}$ is the heuristic information, $\mathbf{di, j}$ is the Mahalanobis distance value of the pixel at the position (i, j) of the image, the function Mahalanobis distance is to compute the similarity between pixel features at i,j with a local group of pixels, and its value relies on the mean, standard deviation and gray value of image pixels values on the 3*3 window.

D. Solution Construction Phase

In each construction step, each ant (h, k) calculates the heuristic information equation (1) to move to pixel *j* from pixel position *i* according to probability function equation (3).

$$P_{ij}^{c,k}(t) = \frac{(\tau^c_{i,j}(t))^{\alpha}(\eta^c_{i,j})^{\beta}}{\sum_{g^c \in N_i^{c,k}} (\tau^c_{i,g}(t))^{\alpha}(\eta^c_{i,g})^{\beta}} \dots\dots\dots (3)$$

Where $\boldsymbol{\tau^c_{i,j}(t)}$ represent the value of pheromone information of the path i to j in colony c; $\mathbf{N_i^{c,k}}$ is the feasible neighbors of pixel(i,j) for the ant k in colony c. the constants $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ represent the influence of pheromone information and heuristic information, respectively. $\boldsymbol{\tau^c_{i,j}(t)}$ give the pheromone quantities (concentration) for a given i,j in iteration number t. $\boldsymbol{\eta^c_{i,j}}$ represents the heuristic information for going from node(i) to node(j) in colony c.

When an ant completes a solution or is in construction phase, the ant will evaluate the solution and will modify the pheromone trail values on the pixel which are used in the solution. Depending on these values, ants deposit a certain quantity of pheromone on the components and thereby, increasing or decreasing the probability of including it in the path. The following ants will use the pheromone information to guide in achieving optimal and better solutions.

E. Pheromone Trail Update Phase

To expand a single ACO pheromone to MAC pheromones, let $\tau^c_{i,j}$ represent the pheromone intensity on the pixel (i, j) in the c-th colony at time t. The intensity of the pheromone on the pixel in each colony is initialized to a small value (positive) $\tau^c_{0,j}$ at time 0. When each ant have completed its tour in **n** time intervals, the pheromone matrix intensity $\tau^c_{i,j}$ becomes

$$\tau^c_{i,j}(t+n) = \rho . \tau^c_{i,j}(t) + \eta^c_{i,j}(t) . \Delta\tau^c_{i,j} \dots\dots\dots\dots\dots\dots\dots\dots\dots (4)$$

Where ρ is a coefficient such that $(1-\rho)$ represents the evaporation rate of the pheromones between time t and t + n as in the original ant algorithm. The value of ρ must be set to a value less than 1 i.e $\rho = 0.01$ to avoid unlimited pheromone accumulation. $\boldsymbol{\eta_{i,j}}$ is the heuristic matrix. The heuristic information will add to ant's memory and will be used for additional steps. The adapted intensity of the pheromone $\Delta\tau^h_{i,j}$ is defined in equation (5):

$$\Delta\tau^c_{i,j} = \sum_{k=1}^{m} \Delta\tau^{c,k}_{i,j} \dots\dots\dots\dots\dots\dots (5)$$

Where $\Delta\tau^{c,k}_{i,j}$ represents the intensity of pheromone is laid by ant (c, k) between time t and t+1, and this is given in equation (6):

$$\Delta\tau^{c,k}_{i,j} = \begin{cases} Q^c \ if \ ant \ k \ in \ colony \ h \ laid \ on \ pixel(i,j) \\ 0 \quad otherwise \end{cases} \dots\dots\dots\dots\dots\dots (6)$$

Here, Q is a constant represent the intensity of the pheromone laid by ant on the current pixel in its tour according to the type of colonies.

F. Global Update

Verify if all ants moved one step in each iteration, if yes; apply the global pheromone update process. The process of global update is performed after the movement of all ants in each construction-step according to equation (7):

$$\tau^{cn}(t) = (1-\psi).\tau^{cn-1}{}_{i,j}(t-1) + \psi.\tau^{c}(0) \quad \ldots\ldots\ldots\ldots\ldots\ldots (7)$$

Where ψ is the *pheromone decay coefficient* and $\tau(0) = \tau^{c}{}_{init}$. The pheromone matrix is updated at this stage with the consideration of the decay coefficient and the pheromone matrix built until now.

IV. IMPLEMENTATION AND RESULTS

The implementation of proposed algorithm use multi-threaded technique to obtain the benefit of the multiple cores processor. Each thread can be scheduled on a different CPU core. In parallel implementation of the proposed algorithm, every colony contains 15 ants process by one core. Core 0 is responsible for initialization, collection and display of the results, while all cores are responsible for constructing solutions to the problem.

1. In master Core we will create a thread that controls the initialization and distribution of parameters and pheromone matrix to each colony.
2. For each slave Core we will create single colony have K ants. The number of slaves are 3 according to text, image and background region in document image
3. Each slave apply typical ACO algorithm for each ant in colony to find each segment.
4. After all the ants in each colony have finished their work, each colony will send the pheromone matrix to the master Core thread.
5. Finally, when master Core thread received the solution from each colony collects solutions and presented as a single solution.

In the experiments performed all the colonies shared the same initial parameters:
α= 1, β= 1, ρ= 0.1, q_0= 0:9 and δ = 0:05. The value $\boldsymbol{\tau_0}$ is set to 0. Each experience has run 10 times. Fig. 5 shows the result of implemented proposed algorithm on input image (image size 690*1144). Fig. 6 the result obtained from each core. The execution time of proposed algorithm to segment the input image is 57.39548 *ms*. The maximum utilization of 4 core shows in Fig 7.

Fig. 5 (A) Original Image (B) Segmented Image (C) Pheromone Matrix

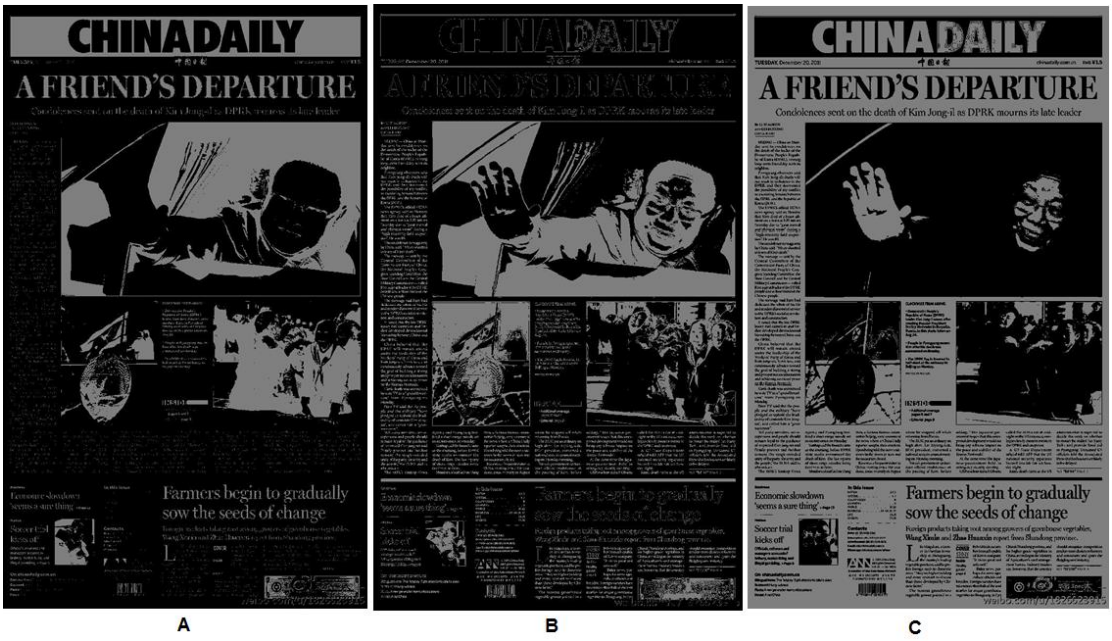

Fig. 6 (A) First Cluster from core2 (B) Second Cluster from 3 (C) Third Cluster from core 4

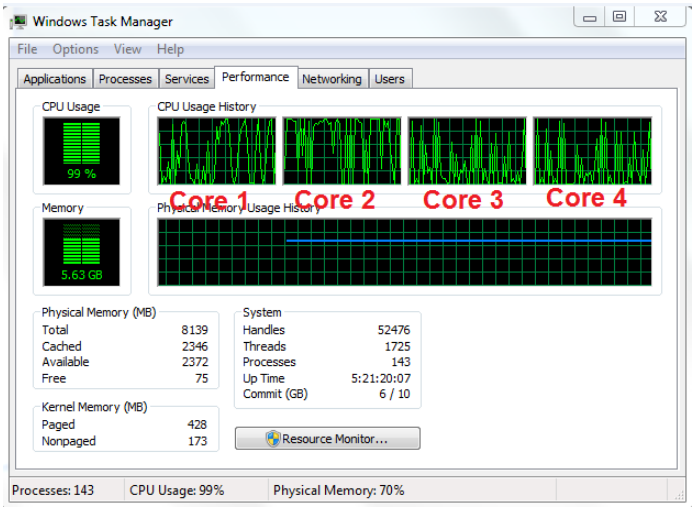

Fig. 7 Cores of implementation platform

In performance evaluation process, parallel and sequential algorithms are all executed 10 times on dataset contain 3 different image implemented on a quad-Core machine, and the results are the average value of the 10 times execution of the two algorithms. Time consumption comparison of these two algorithms is listed in TABLE I.

TABLE II
Execution Time for Sequential ACO and Proposed MACO

Image No.	Size	Execution Time (Sequential algorithm)	Execution Time (Sequential algorithm)
1	660*396	59.4397 *ms*	23.6169 *ms*
2	800*1035	169.2554 *ms*	52.3954 *ms*
3	700*980	133.1902 *ms*	46.2333 *ms*

From the experimental results, the proposed algorithm performs better than sequential algorithm and has great speedup (the solution precision is also guaranteed).

V. CONCLUSION

To effectively exploit the H/W capabilities for proposed algorithm, generate a thread at the application level, consider a good idea also execute separated threads on cores in multicore CPU to improve the performance. Therefore, using a threading technique considers best practice to maximize scalability power processing of the multicore processors. At each time, only one thread of application can runs on one CPU core. Therefore, application have 4 threads will run faster on 4 core processor. In this paper, concept of multi core multithread processing is used to enhance the performance of the ACS algorithm.

The given parallelized ACO algorithm enhances the behavior of the system and reduces the computational time to a certain extent. The computational time is reduced by parallelizing the ACO section of the algorithm by multi-core parallel programming, exploiting the rapid development of multi-core architecture and hence, reducing the computation time considerably.

REFERENCES

[1] Jong, J. and Wiering, M., *Multiple Ant Colony System for the bus-stop Allocation Problem*, In Thirteenth Belgium-Netherlands Conference on Artificial Intelligence (BNAIC'01), 141–148, 2001.

[2] Kawamura, H., Yamamoto, M., Suzuki, K. and Ohuchi, A., *Multiple Ant Colonies Algorithm Based on Colony Level Interactions*, IEICE Trans. Fundamentals, E83-A(2), 2000.

[3] M. Dorigo and G. Di Caro, *The Ant Colony Optimization Meta-Heuristic*, Advances Topics in Computer Science, McGraw-Hill, London, pp. 11–32, 1999.

[4] M. Dorigo and T. Stutzle, *Ant Colony Optimization*, MIT Press, Massachusetts, 2004.

[5] J. Li, X. Hu, Z. Pang, and K. Qian, *A parallel Ant Colony Optimization Algorithm Based on Fine-grained Model with GPU acceleration, International Journal of Innovative Computing, Information and Control*, vol. 5, pp. 3707–3716, 2009.

[6] L. Chen, H.-Y. Sun, and S. Wang, *Parallel Implementation of Ant Colony Optimization on MPP*, in *Machine Learning and Cybernetics, International Conference on*, vol. 2, pp. 981 –986, 2008.

[7] Su, S.Q. and Liang, S.Z. *Parallel Study of Ant Colony Algorithm*, Jisuanji Yu Xiandaihua, 10,18–21,(In Chinese), 2009.

[8] Randalla, M. and Lewisb, A. *A Parallel Implementation of Ant Colony Optimization*, J. Parallel Distrib. Comput. 62, 1421–1432, 2002.

[9] B. Bullnheimer, G. Kotsis, and C. Strauss, *Parallelization Strategies for the Ant System*, In R. De Leone, A. Murli, P. Pardalos, and G. Toraldo, editors, High Performance Algorithms and Software in Nonlinear Optimization, volume 24 of Applied Optimization, pages 87–100. Kluwer, Dordrecht, 1997.

[10] T. Stützle, *Parallelization Strategies for Ant Colony Optimization*, Fifth International Conference on Parallel Problem Solving from Nature (PPSN V), volume 1498, pages 722–731. Springer-Verlag, New York, 1998.

[11] R. Michel and M. Middendorf, *An Island Based Ant System with Lookahead for the Shortest Common Subsequence Problem*, In Fifth International Conference on Parallel Problem Solving from Nature,'' Vol. 1498, pp. 692–708, Springer- Verlag, Berlin (1998)

[12] M. Pedemonte, S. Nesmachnow, and H. Cancela, *A survey on Parallel Ant Colony Optimization*, Applied Soft Computing, 11:5181–5197, 2011.

[13] B. Scheuermann, K. So, M. Guntsch, M. Middendorf, O. Diessel, H. ElGindy, and H. Schmeck, *FPGA Implementation of Population-Based Ant Colony Optimization*, Applied Soft Computing, 4:303–322, 2004.

[14] B. Scheuermann, S. Janson, and M. Middendorf, *Hardware-Oriented Ant Colony Optimization*, Journal of Systems Architecture, 53:386–402, 2007.

[15] A. Catala, J. Jaen, and J. Mocholi, *Strategies for Accelerating Ant Colony Optimization Algorithms on Graphical Processing Units*, In IEEE Congress on Evolutionary Computation, pages 492–500. IEEE Press, 2007.

[16] J. Wang, J. Dong, and C. Zhang, *Implementation of Ant Colony Algorithm Based on GPU*, In E. Banissi, M. Sarfraz, J. Zhang, A. Ursyn, W. C. Jeng, M. W. Bannatyne, J. J. Zhang, L. H. San, and M. L. Huang, editors, Sixth International Conference on *Computer Graphics, Imaging and Visualization: New Advances and Trends*, pages 50–53. IEEE Computer Society, 2009.

[17] A. Delévacq, P. Delisle, M. Gravel, and M. Krajecki, Parallel Ant Colony Optimization on Graphics Processing Units, Journal of Parallel and Distributed Computing, page doi :10.1016/j.jpdc.2012.01.003, 2012.

[18] M. Randall and A. Lewis, *A parallel implementation of ant colony optimization*, *Jounal of Parallel and Distributed Computing*, 62:1421–1432, 2002.

[19] B. Bullnheimer, G. Kotsis and C. Strauss, *Parallelization Strategies For Ant System*, in: R. DeLeone, A. Murli, P. Pardalos, G. Toraldo (Eds.), High Performance Algorithms and Software in Nonlinear Optimization, Kluwer Series of Applied Optimization, vol. 24, Kluwer Academic Publishers, Dordrecht, Netherlands, pp. 87–100, 1998.

[20] M. Middendorf, F. Reischle and H. Schmeck, *Information Exchange in Multi Colony Ant Algorithms*, in: J. Rolim (Ed.), Parallel and Distributed Computing, 15 IPDPS 2000 Workshops, Third Workshop on Biologically Inspired Solutions to Parallel Processing Problems (BioSP3), LNCS, vol. 1800, Springer-Verlag, Berlin, pp. 645–652, 2000.

[21] M. Middendorf, F. Reischle and H. Schmeck, *Multi Colony Ant Algorithms*, Journal of Heuristics: Special Issue on Parallel Metaheuristics 8 (3) ,305–320, 2002.

[22] S.C. Chu, J.F. Roddick and J.S. Pan, *Ant Colony System With Communication Strategies*, Information Sciences 167 ,63–76, 2004.

[23] M. Rahoual, R. Hadji and V. Bachelet, *Parallel Ant System for the Set Covering Problem*, in: M. Dorigo, G. Di Caro, M. Sampels (Eds.), Ants Algorithms – Proceedings of ANTS 2002, Third International Workshop on Ant Algorithms, Lecture Notes in Computer Science, vol. 2463, Springer-Verlag, Heidelberg, Germany, pp. 262–267, 2002.

[24] E. Talbi, E. Roux, C. Fonlupt and D. Robillard, *Parallel Ant Colonies for Combinatorial Optimization Problems*, in: J. Rolim et al. (Eds.), Parallel and Distributed Processing 11 IPPS/SPDP'99 Workshops, LNCS, vol. 1586, Springer, Berlin, pp. 239–247, 1999.

