Optimization of the Running Speed of Ant Colony Algorithm with Address-based Hardware Method

ElnazShafighFard, Khalil Monfaredi
Engineering Faculty, Department of Electrical and Electronic Engineering, Azarbaijan Shahid Madani University, Tabriz, Iran
shafighfard@azaruniv.edu

***Abstract*-Ant colony algorithm is an algorithm inspired by the Nature. It has been used a lot for solving complex issues and finding optimum answers. However, this algorithm is problematic due to its huge calculations, resulting in the decrease of its running speed. Such a decrease is considered a weak point for the much used algorithm. This paper presents an optimized core design purely based on hardware technology. By presenting a special algorithm which runs on a programmable chip based on nodes' address in memory, the repetition of the same function is avoided. Assessments done on ISE Xilinx area have optimized the speed of the suggested Ant colony algorithm running time compared to the hardware method based on the population for 27 times, method based on ID 17.74 times, and the compound hardware-software method up to 15.71 times.**

***Index Terms*-ant colony, hardware technology, speed of process, address, speed**

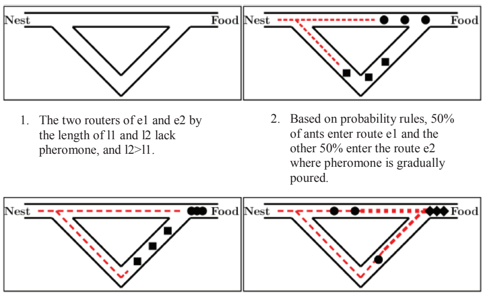
I. INTRODUCTION

Getting inspiration from the Nature and imitating it is one of the common and prevalent methods in artificial intelligence. Despite its wide application, this method encounters a serious problem due to its low running speed. Evolutional Algorithm and calculations that offer solutions to problems are made of a cluster of methods, which are inspired by the evolution of animals, plants and insects from the Nature. These algorithms, which are used for obtaining optimum answers, have the disadvantage of complex and time consuming calculations. The problem of running speed was not addressed until the beginning of 1990s when the running speed of ant colony algorithm was optimized by parallel software methods [1][2]. Until the year 2002, all procedures were based on software, and they mostly used parallel processing methods [3][4]. In some cases, these algorithms were run in parallel on a graphic processor made of several cores [5]. During the 2000s, some methods for the implementation of hardware on a reconfigurable chip were offered by Dr. Sherman et al [6], and then a work on the implementation with the CMOS technology was presented [7]. Meanwhile, two projects were offered based on reconfigurable chip, using all available IPs and cores in it [8][9]. Finally, in 2012, a project which used a combination of software and hardware methods enhanced the hardware running speed and the software flexibility [10]. In this paper, the first part will introduce the evolutional algorithms, particularly the ant colony algorithms. In the second part, the method will be delineated. In the third part, evaluation and simulation will be carried out, and a comparison will be made with the previous methods. In the fourth and the last part, the conclusion and future work will be presented. Characteristics such as heterogeneity, autonomy, scalability, adaptability and resources computation-data separation, which make the load balancing problem more difficult, will be highlighted in the paper as well.

II. ANT COLONY ALGORITHM

Ant Colony System (ACS) [6] is one of the most successful algorithms used in combinatorial optimization problems, such as the Traveling Salesman Problem (TSP). The algorithm is inspired by the foraging behavior of a colony of ants when they communicate with each other.

In a community, a group of members cooperates with each other to reach a certain ultimate goal. This method of cooperation is more beneficial than when the members act individually. An ant colony can be defined as an organization of agents that cooperates with each other using pheromone, and exchanges information based on pheromone update. The calculation of collective intelligence is inspired by the behavior of some animals like ants, termites, bees, fish and groups of birds. In such structures, each individual carries out a very simple act; however, the cooperation between these insects presents a complex behavior. The collective behavior of a community is made of a combination of its individuals' behaviors in a non-linear fashion. In other words, there is a complex relation between the individual behavior and collective behavior in a community. The collective behavior is also subjected to the relation between individuals since the cooperation is the result of the accumulation of agents' experience, which is also the result of the progress of the community. The procedure of finding the shortest path by ant colony is clearly shown in Figure 1, where in the standard mode, every model of ant colony could be shown in this way: G=(V,E) in which V stands for nodes and E stands for the space between two nodes. V itself includes VS and Vd; S to d is the Source to Destination path. These two are commonly called the nest node and the food node, respectively. Further, E includes e1 and e2 links, for each of which the lengths of L1 and L2 are considered, where L2>L1. The probability rules based on pheromone variable are used to choose one of the roots of e1 or e2.



1. The two routers of e1 and e2 by the length of l1 and l2 lack pheromone, and l2>l1.
2. Based on probability rules, 50% of ants enter route e1 and the other 50% enter the route e2 where pheromone is gradually poured.
3. The ants which were on the route e1, with lesser length, found food, also the volume of pheromone in the route e1 is more than that of the route e2
4. Ant was returning from e1, and the quantity of injected pheromone in the route e1 was far more than that of the route e2, so the probability of choosing route e1 in future is a lot since more pheromone has been produce there.

Figure 1: The procedure of finding the shortest path by the ants

When an ant is in the city i and wants to go to the next city, for example city j, it uses distributional probability to choose the next city [11]:

$$p_{ij} = \frac{[\tau_{i,j}]^{\alpha}[\,\eta_{i,j}]^{\beta}}{\sum[\tau_{i,j}]^{\alpha}[\,\eta_{i,j}]^{\beta}} \qquad (1)$$

However, for distributional probability in Eq.1, there should be a link between i and j, so that i can be a neighbor of j. As it is evident in the above formula, the js should belong to the assembly of N which includes the nodes that have not been selected before since the already selected pijs are considered zero. After the calculation of the pijs, q_0, which is in the span of (0, 1], is considered as a variable. It is then compared with q, which is one of the parameters of ant colony algorithm. Thus, the following Eq.2 is for the movement from node r to node s through the route u, if $q_0>q$:

$$P_k(r,s) = \begin{cases} \frac{[Ph(r,s)].[D(r,s)]^{\beta}}{\sum_{uc_{jk}(t)}[Ph(r,u)].[D(r,u)]} \\ 0 \end{cases} \qquad (2)$$

In this case, the city s will not be chosen.

If $q_0<q$, then:

$$s = \begin{cases} Arg\max\{[Ph(r,s)].[D(r,s)]^{\beta}\} & \text{if the best configuration is selected} \\ s & \text{if the next configuration is selected} \end{cases} \qquad (3)$$

Once the ant has searched for one route and finished one repetition, the final update, which is sometimes called the general update, is carried out. In this episode, every ant that has built the shortest route is allowed to increase the pheromone of the manes in his route. The increase in the pheromone is delineated in the following formula:

$$(\tau_{ij\ new}) = (1\text{-}\rho)\tau_{ij\ old} + (\rho\Delta\,\tau_{ij}) \qquad (4)$$

In this update, only the shortest route of the pheromone of the manes is modified, and more pheromone is allocated to the manes with shorter lengths.

III. Method

In this section, a framework is offered for ant colony algorithm modeling which uses a hardware design based on a system on programmable chip that broadly covers various applications. A software result on a desktop processor was compared with the architectural modeling results based on a system on programmable chip, which was also address-based. The purpose of the present method is to decrease the processing time of the algorithm. The proposed architecture is concerned completely with hardware, and it is address-based.

The efficiency of the proposed architecture is evaluated by several criteria. Operations are done by various tools like Xilinx, ISE, MaxPlus II and Hardware Description Language (VHDL). A complete software model has been simulated in Matlab.

A. Ant Colony Parallel Algorithm on Hardware

In this section, we will describe the function of ants on the reconfigurable chip's bed consistent with the proposed idea. Figure 2 shows the manner of ant1, in which its cost function is compared to other cost function of ants.

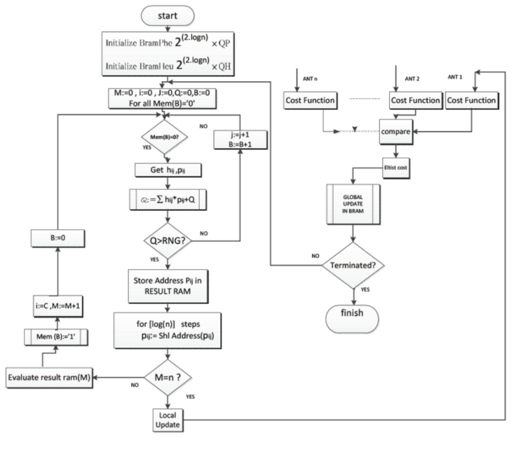


Figure 2: Proposed flowchart for one ant (ant 1) in hardware bed

As shown in the flowchart, first, two ram blocks are given values. One of these blocks is for storing the pheromone's information between two nodes, and the other one, which is in the rom despite its placement in ram, is used for storing the heuristic coefficients. By taking this action, no modification is made between the two nodes during the run time of the program. More importantly, the Result memory that controls which nodes should be chosen, has one byte home for each node. These homes are set to zero at the beginning of the process, implying that no node is chosen. When a node is chosen, the related field is set to one. In all ant colony algorithms, which run on the hardware bed, the value of stored Pheromone in the memory is consider as n*n matrix, as it is shown in Figure 3.

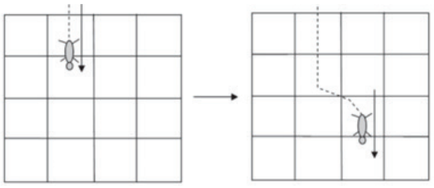


Figure 3: The movements of ants in pheromone matrix

Here, the values of column and line i, j refer to the pheromone matrix in which a line is allocated to each ant. In the related flowchart, ant No.1 is first allocated to the columns and line zero. It then starts its search, and reads the values related to the pheromone and heuristic coefficient for i, j cell from the related memory. The ant acts according to the flowchart and, by choosing each node, the value of M which has been set to zero at the starting point is increased by one. The address for each selected node is stored in a memory called Result, and the ant finds its next line from the currently found node. Thus, during a function of shift to the left, which is a substitute for multiplication function in order to decrease the consumption potency, the ant moves from one line to another during every move. But if a node is not qualified as being greater than the random value created by LFSR method [15], the ant will just change its column and move to the next one and stay there until the a node is selected. When a node is selected, the ant checks to see if M=n, showing whether all nodes have been selected or not. If M equals to n, the phase of finding a solution for that repetition will be closed, then the local update phase, and the phase for evaluation of the efficiency of ants will follow. By comparing the cost function of all the ants, the function with the least cost will be selected, hence the pheromone memory will be updated. After updating, as it was explained in the introduction to ant colony algorithm section, this algorithm can be repeated infinitely. Once all phases are finished, the requirements for repetition are checked. On the other hand, if the requirement for terminating the operation is met, the process of running the program will be closed.

B. Ant Colony Optimizing Algorithm's Architectural Framework based on System on Programmable Chip

An improvement is made in algorithm in order to decrease the hardware cost. As it was shown in the first section, the direct operation has been deleted in this algorithm. Further, an attempt has been made to use the shift register operation instead of the multiplication operation. To ensure that the program would not be defunct, a set of simple modifications has been made in updating the part so that the values would not be a decimal.

C. Structure of the Architecture

The designed framework for the algorithm is made of a reconfigurable chip. As a result, the definitions of the colony parameters are stored in a two-block memory on a reconfigurable chip, and all algorithm calculation operations are modeled on FPGA logics based on the hardware. Figure 4 shows the proposed frameworks, and the connections between different blocks. The figure shows the following: two independent memories, city selection unit which has some sub-blocks, evaluation unit, and updating unit. These parts will be discussed at length in the following section.

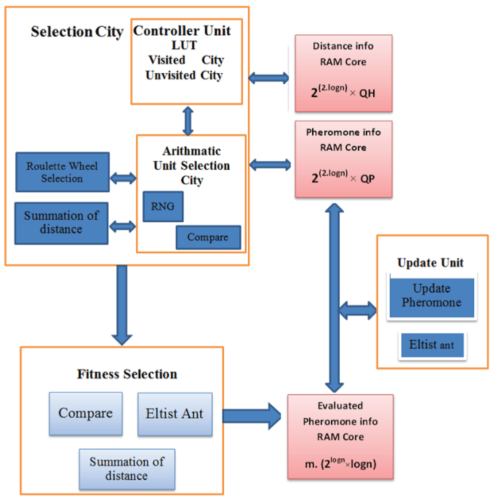


Figure 4: The structure of ant colony algorithm architecture based on a system on programmable chip

The memories have been selected from inside the chip, and the hardware core of the ant colony optimizing algorithm has been modeled on FPGA logics.

IV. Evaluation of the Proposed Method

As shown in the Figure 5, when ants repeat the process of searching and finding an optimum solution for 20 times, a time of 20777.5 ns is spent.

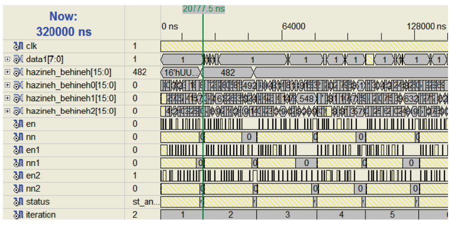
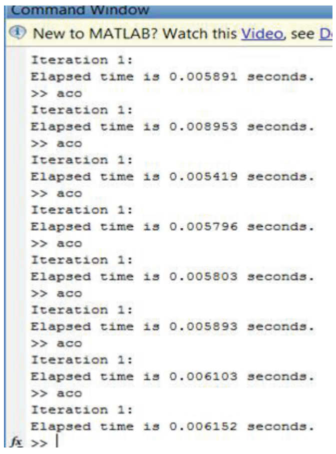


Figure 5: The result of the simulation

This simulation was also run in the same manner in MATLAB software with 16 nodes and 3 ants, as shown in Figure 6. In order to increase the precision, the experiment was repeated for 8 times.



```
Iteration 1:
Elapsed time is 0.005891 seconds.
>> aco
Iteration 1:
Elapsed time is 0.008953 seconds.
>> aco
Iteration 1:
Elapsed time is 0.005419 seconds.
>> aco
Iteration 1:
Elapsed time is 0.005796 seconds.
>> aco
Iteration 1:
Elapsed time is 0.005803 seconds.
>> aco
Iteration 1:
Elapsed time is 0.005893 seconds.
>> aco
Iteration 1:
Elapsed time is 0.006103 seconds.
>> aco
Iteration 1:
Elapsed time is 0.006152 seconds.
fx >> |
```

Figure 6: The simulation in MATLAB

The average time for one repetition in 8 software simulation experiments was 0.0002301 seconds. Thus, in order to compare the running time of simulation of algorithm in software and hardware methods in a frequency of 50 MHz, the running speed of software method is divided by the running speed of hardware method and the result is agglomerated. The result shows that the running speed of hardware methods is 110 times more than that of the software method. Besides the suitable architecture, the reason for such a result is the inherent efficiency of the parallel processing of algorithm.

The method and the result are agglomerated. The result in Table 1 shows that the running speed of hardware methods is 110 times more than that of the software method. Besides the suitable architecture, the reason for such a result is the inherent efficiency of the parallel processing of algorithm.

Table 1
Comparison of Optimization of Software Speed with System on Chip Architecture

The name of the method	Software speed	Hardware speed	The amount of optimization
Architecture for high speed ant colony optimization	334.0 s	53.7 s	6.2 times
A combination of software-hardware	0.7380 s	0.01440 s	7 times
Population-based ant colony optimization on FPGA			2-4 times based on the change of ants
Proposed method	0.0002301s	20777.5 ns	110 times

V. Conclusion and Future Works

In this paper, an addressed-based modeling of ant colony optimizing algorithm based on system on chip was presented. This design has the utility as an optimizer. The architecture of the proposed design utilizes a series of pipeline and parallel structures which enable it to optimize various applications by ant colony algorithm. The presented work that used hardware method increased the running speed of ant colony algorithm to 15.71 times more than that of the last reported work (A combination of software-hardware).

Another interesting work is to present a hardware-software combinational method with the proposed hardware core to increase the flexibility of algorithm.
One of the future works that can be done in this field is to make the proposed work scalable. The job can be executed by a network on chip technology.

References

[1] M. Dorigo, “Optimization, Learning and Natural Algorithms,” *PhD thesis*, Politecbico di Milano, Italy, 1992.
[2] M. Dorigo, “Parallel Ant System: An Experimental Study,” *Unpubished Manuscript*, cited by M. Dorigo, 1993.
[3] M. Pedemonte, S. Nesmachnow, H. Cancela, “Survey on Parallel Ant Colony Optimization,” *Applied Soft Computing Journal*, 2011.
[4] P. Delisle, M. Gravel, M. Krajecki, C. Gagné, W. Price, Comparing Parallelization of an ACO: Message Passing vs. Shared Memory, *Proceedings of the 2nd International Workshop on Hybrid Metaheuristics,* Lecture Notes in Computer Science vol. 3636, 2005.
[5] J. Fu,L. LEI,G. Zhou, “A Parallel Ant Colony Optimization Algorithm with GPU Acceleration based on All-in-roulette Selection,” *Proceedings of the 3rd International Workshop on Advanced Computational Intelligence* , 2010, pp 260-264.
[6] B. Scheuermann, S. Janson, M. Middendorf, “Hardware-oriented Ant Colony Optimization,” *Journal of Systems Architecture* 53 (7) (2007) 386–402.
[7] K. Gheysari, Khoei. A, Mashoufi .B, “High Speed Ant Colony Optimization CMOS Chip”, *Expert Systems with Applications* pp 3632-3639 (2011).
[8] M. Yoshikawa and H. Terai, “Architecture for High Speed Ant Colony Optimization,” *Proceedings of IEEE International Conference on Information Reuse and Integration*, Las Vegas, NV, 1-5 .
[9] M. Yoshikawa and H. Terai, “Hardware-oriented Ant Colony Optimization Considering Intensification and Diversification,” *Advances in Greedy Algorithms*, I-Tech, 359-68.
[10] Li. S, Hao Yang. M ,Wei WENG. CH,Hong Chen. Y, “Ant Colony Optimization Design and Its FPGA Implementation,” *IEEE International Symposium on Intelligent Signal Processing and Communication System,* PP 262-265, November 4-7, 2012.
[11] M. Dorigo, G. Di Caro, L. Gambardella, “Ant Algorithms for Discrete Optimization,” *Artificial Life 5 (2)* (1999) 137–172
[12] D. Merkle, M. Middendorf, “Fast Ant Colony Optimization on Runtime Reconfigurable Processor Arrays,” *Genetic Programming and Evolvable Machines 3 (4)* (2002) 345–361
[13] H. Bai, D. OuYang, X. Li, L. He, H. Yu, “Max-min Ant System on GPU with CUDA,” *Proceedings of the 2009 Fourth International Conference on Innovative Computing, Information and Control*, IEEE Computer Society, 2009, pp. 801–804.
[14] H. Duan, Yaxiang. Yu, Jie .Zou and Xing. Feng, “Ant Colony Optimization-based Bio-inspired Hardware.
[15] P. Martin, “An Analysis of Random Number Generators for A Hardware Implementation of Genetic Programming using FPGA and Handek-C,” *Proc. Genetic and Evolutionary Computation*.