Ant Colony Optimization Scheduling

Jaouaher Belgacem
Department of Electronic Engineering
Hochschule of Hamm-Lippstadt
Lippstadt, Germany
jaouaher.belgacem@stud.hshl.de

***Abstract*—**
***Index Terms*—ACO, Hardware/Software Codesign, Ant colony Optimization**

I. Introduction

In 1991, Marco Dorigo and his co-workers presented the Ant Colony Optimization (ACO) after studying and observing the behaviour of real ants in nature. In solving optimization problems, the ACO algorithm performs better than other algorithms. Individuals and feedback of information are essential to the ACO algorithm, where the ants are collaborating in an organized way through the pheromone in a polyphony. Hence, optimization-related problems can be solved by imitating the ants' behaviour [16] [2].

In the recent couple of years, hardware elements are getting complicated especially as they have a predominant component that includes a hardware platform which executes software programs. Thus, Hardware/Software Codesign means meeting those two parts together. Hence, researchers have been trying to use different optimization methods to maximize the performance of the circuits. Thus, ACO has shown good optimization results in solving optimization-related problems such as the travelling salesman problem, job-shop scheduling problem, graph colouring problem and so on [2] this high performance is achieved due to the meta-heuristic feature that ACO has, by C´combining a prior information about a potential solution with a posteriors information about previously successful solutions.

In order to escape local optima, meta-heuristic algorithms rely on some basic heuristics to help them escape them. In a constructive heuristic, elements are added to build a good complete solution from a null solution, in a local search heuristic, some elements of a complete solution are modified iteratively to make it better. In spite of iterating, the meta-heuristic part enables the low-level heuristic to find better solutions than alone.

A lot of challenging problems could face the designer like partitioning or scheduling during high-level synthesis [**?**] [18] [17]. In the first section, we will discuss different terminologies related to the topic, in section two we will provide a deep explanation of how the ant colony optimization is working and in the last section we will show how this algorithm could be integrated to solve optimization challenges related to High-level synthesis.

II. Background

A. High Level Synthesis

The automated process of converting an algorithm into a dedicated digital circuit is called high-level synthesis or in other words architectural synthesis [11]. HLS concept is used to produce an efficient Electronic System-level design on a hardware and software level [12]. Thus, it is responsible for decreasing the production time and verification time as well as facilitating the flow of the power analysis and as a consequence reducing the production costs [11] [12]. This High-level synthesis design is performed in different steps as shown in Figure 1: The first step of the HLS is identifying the

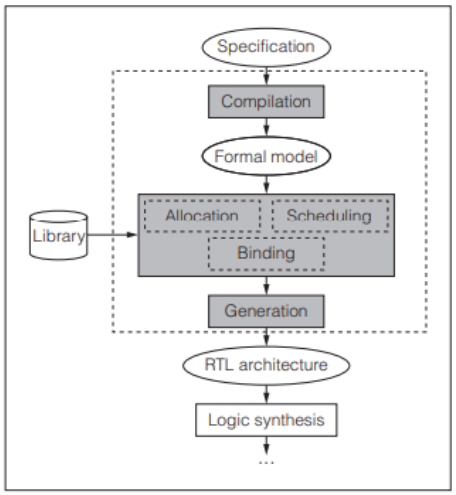

Fig. 1. The different steps of HLS design [12] [7]

functional specifications .... to be continued There are different HLS tools that are used for

B. Scheduling

C. Hardware Software Co-design

The task of designing an embedded system's architecture is a complicated task as it is composed of different elements such as hardware and software. Thus, to make this task more flexible and relatively more straightforward, researchers proposed an appealing design technique which is called Hardware/Software Co-design [14]. The traditional way to define Hardware/software co-design is about putting the effort of designing hardware elements in a collaborating way in a

single design effort [15]. This approach aims to merge the different design processes which are namely hardware and software designs. Where, the software design utilizes temporal decomposition and is well suited for flexibility, while hardware design utilizes spatial decomposition. Hence, It is possible to get solutions that are both flexible and effective when hardware and software are combined successfully [14].

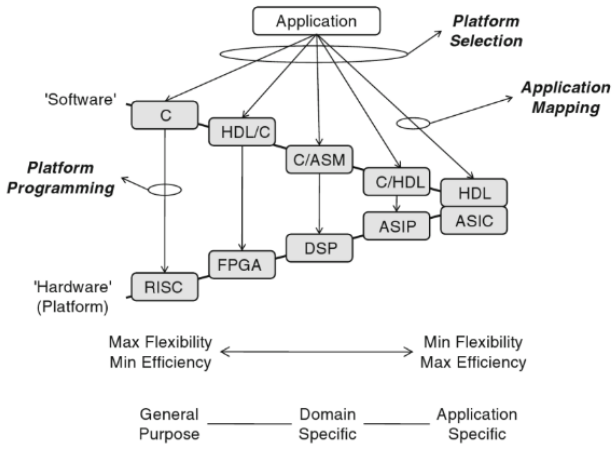

Fig. 2. Hardware/Software Codesign space [15] [7]

In the figure above, we can illustrate some examples of programmable devices such as FPGA, ASIP, DSP, RISC and ASIC and the software is mapped to the hardware platforms using programming them. Besides, Application mapping means writing software programs using suitable programming languages for the aimed platform [15].

III. Ant Colony Optimization Algorithm

The ACO concept is that the ants have to find a way to reach their target which is mainly reaching the food that is placed on the other side of their nest. The ants are guided while exploring the path with the help of the pheromone which is a special chemical trail [16] [9]. As shown in the figure, ants

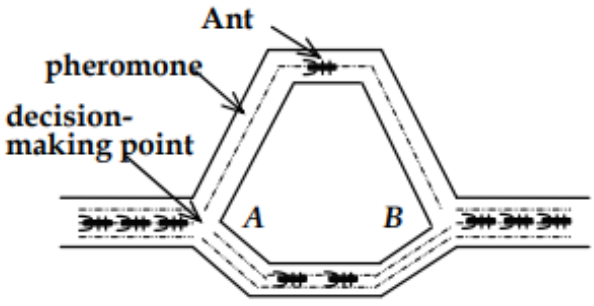

Fig. 3. Ants choose their path according to the pheromone [16]

have to decide on the path they will follow once they reach the decision node A. There are two paths, one is longer than the other. Since the ants have the same crawling speed, the group of ants who choose the shortest path will reach destination node B first. It is important to note that, the choice is made randomly [16].

Mainly, the ACO algorithm is based on a number of iterations. For each iteration, several ants use heuristic information as well as previous populations' experience to construct complete solutions [2]. Below is a simple pseudo-code that illustrates a meta-heuristic approach to ant colony optimization, which can be applied to a range of optimization problems [1]:

ACO meta-heuristic Algorithm
Set parameters, initialize pheromone trails
while termination condition not met **do** ConstructAntSolutions
ApplyLocalSearch optional
UpdatePheromones
endwhile

The three main components of the algorithm are:

ApplyLocalSearch: After the construction of solutions, it is a common practice to enhance the solutions generated by the ants through a process known as local search, before updating the pheromone. This particular phase, which is tailored to the specific problem at hand, is not obligatory but is typically incorporated into advanced ACO (Ant Colony Optimization) algorithms [1].

ConstructAntSolutions: A group of m simulated ants creates solutions using elements from a finite set of available solution components $C = c_{ij}$, where i ranges from 1 to n and j ranges from 1 to $|D_i|$. The construction of a solution begins with an empty partial solution sp. During each construction step, the partial solution sp is expanded by adding a feasible solution component from the set N(sp), which is defined as the collection of components that can be incorporated into the current partial solution sp without violating any of the constraints present.

UpdatePheromones: The objective of updating the pheromone is to amplify the pheromone values linked to favourable or promising solutions while reducing those associated with unfavourable ones. This is typically accomplished by (i) diminishing all pheromone values through pheromone evaporation, and (ii) elevating the pheromone levels connected to a selected group of desirable solutions.

A. Explaining ACO through TSP example

At first, the ACO was applied to solve the Travelling Salesman Problem (TSP) which is composed of a set of cities whew the distances between the different cities are known. The ultimate goal was to find the shortest Hamiltonian tour on a fully connected map, where each city is allowed to be visited only once. Each edge of the graph represents the cities' connection and every vertex indicates the city itself. Besides, Ants can read and modify a variable called a pheromone associated with each edge [1].

The TSP problem is a famous optimization challenge that we use as an application to illustrate how the ACO is used to overcome this challenge. Let N be the number of towns, let i be the city edge and city j. Let the length d_{ij} be the distance between cities i and j to each edge(i,j). One of the

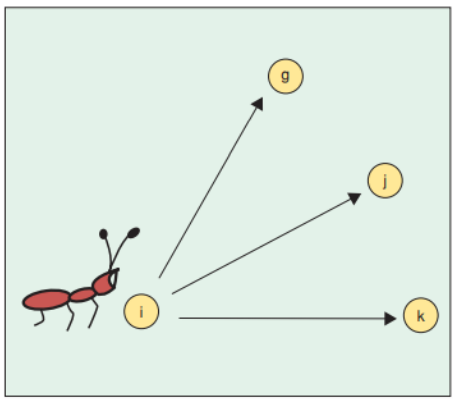

Fig. 4. The ant in city i decides which city to visit next by using a stochastic process. If j has never been visited before, it will be selected with a probability proportional to the pheromone associated with edge (i,j). [1]

best ACO algorithms is the Max-Min Ant system (MMAS) [16] The algorithm is described using different rules such as:

- **The rule of updating Pheromone:** While the ants are moving toward the destination node, they should leave their pheromone on the edges. Every completed trip by the ants is considered an iteration. Thus, the sum pheromone of one boundary is determined by the following equation [16]:

$$\tau_{ij}(t+1) = \Delta\tau_{ij} + (1-\rho)\tau_{ij}(t) \quad (1)$$

Where 1-ρ represents the persistence rate of the prior pheromone and $\rho \in (0,1)$. It represents the evaporation rate of the pheromone. However, in MMAS, only the best updates the trails of the pheromone which will give the value of [16]:

$$\tau_{ij}(t+1) = [\Delta\tau^b_{ij}est + (1-\rho)\tau_{ij}(t)]_t au_m in^t au_m ax \quad (2)$$

where tau_max and tau_min represent respectively the upper and lower bounds of the pheromone and the $\Delta\tau^b_{ij}est$ is [16]:

$$\Delta\tau^b_{ij}est = \begin{cases} [1/L_b estif(i,j) belongstothebestpath \\ else \\ 0 \end{cases} \quad (3)$$

Let L_best be the cost of the best solution achieved or the best solution until a specific iteration and it could be a mix of both [16].

- **The Ants' movement rule:** The transition of the ants from one city to another is happening randomly. Hence, The cities that have been entered by the ants have to be registered in a table. However, the set of cities that have never been accessed of the *K*th ant must be defined as $allowed_k$ and the next step is to determine the visible degree: η_{ij}, $\eta_{ij} = 1/d_{ij}$. Thus, the probability of *K*th ant choosing a specific city is determined by [16]:

$$P^k_{ij}(t) = \begin{cases} \frac{[\tau_{ij}(t)]^\alpha [eta_{ij}]^\beta}{\sum_{K\in allowed_k}[eta_{ik}(t)]^\alpha [eta_{ik}]^\beta} \\ else \\ 0 \end{cases} \quad (4)$$

Where the most important variables are α and β which determine the relative influence of the trail pheromone and the heuristic information [16]. However, we can achieve a better move with MMAS version of ACO which will be as follows [16]:

$$j = \begin{cases} \arg_{k\in allowed_k}\{\tau_{ij}(t)[eta_{ik}]^\beta\} if p \le p_0 \\ else \\ J \end{cases} \quad (5)$$

where p is an arbitrary number in [0,1]. Hence, with the probability of $0 \le p_0 \le 1$ and based on the pheromone trail and heuristic information, make the best move. However, according to the random variable J with the distribution given by the rule (4) (biased exploration), the move is made with the probability $1 - p_0$ [16].

- **The Initialization of the Pheromone Trail:** The $\tau_max = 1/((1-\rho)C_{nn})$, $\tau_min = \tau_max/(2N)$, and initial pheromone values $\tau_{ij}(0) = \tau_max$ are set at the beginning of the run. C_{nn} is the length of a trip generated by the nearest neighbour heuristic and N represents the number of cities [16].
- **The Stopping rule**: To stop the ants from travelling there are two conditions, which are we reach the limited iteration number, CPU time limitation or the best solution is found [16].

IV. Integration Ant Colony Optimization and the High-Level Synthesis

- explain the HLS - talk about the different applications of ACO

-
-
-

A. Conclusion

Acknowledgment

References

References

[1] Dorigo, Marco, Mauro Birattari, and Thomas Stutzle. "Ant colony optimization." IEEE computational intelligence magazine 1.4 (2006): 28-39.

[2] Deng, Wu, Junjie Xu, and Huimin Zhao. "An improved ant colony optimization algorithm based on hybrid strategies for scheduling problem." IEEE access 7 (2019): 20281-20292.

[3] Kopuri, Shekhar, and Nazanin Mansouri. "Enhancing scheduling solutions through ant colony optimization." 2004 IEEE International Symposium on Circuits and Systems (IEEE Cat. No. 04CH37512). Vol. 5. IEEE, 2004.

[4] Makhoul, Rim. Deep Reinforcement Learning for Resource Constrained HLS Scheduling. Diss. Lebanese American University, 2022.

[5] Al Bataineh, Ali, Amin Jarrah, and Devinder Kaur. "High-speed FPGA-based of the particle swarm optimization using HLS tool." International Journal of Advanced Computer Science and Applications 10.5 (2019).

[6] Scheuermann, Bernd, et al. "FPGA implementation of population-based ant colony optimization." Applied Soft Computing 4.3 (2004): 303-322.

[7] Ferrandi, Fabrizio, et al. "Ant colony optimization for mapping, scheduling and placing in reconfigurable systems." 2013 NASA/ESA Conference on Adaptive Hardware and Systems (AHS-2013). IEEE, 2013.

[8] Brian Dalay. Accelerating system performance using sopc builder. In Proceedings. 2003 International Symposium on System-on-Chip (IEEE Cat. No. 03EX748), pages 3–5. IEEE, 2003

[9] Giovanni De Micheli. Synthesis and optimization of digital circuits. Number BOOK. McGraw Hill, 1994.
[10] Shih-An Li, Min-Hao Yang, Chung-Wei Weng, Yi-Hong Chen, Chia-Hung Lo, and Ching-Chang Wong. Ant colony optimization algorithm design and its fpga implementation. In IEEE International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS 2012), 2012.
[11] Elie Torbey and John Knight. High-level synthesis of digital circuits using genetic algorithms. In 1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No. 98TH8360), pages 224–229. IEEE, 1998
[12] Coussy, Philippe, et al. "An introduction to high-level synthesis." IEEE Design & Test of Computers 26.4 (2009): 8-17.
[13] Cardoso, João Manuel Paiva, José Gabriel de Figueired Coutinho, and Pedro C. Diniz. Embedded computing for high performance: Efficient mapping of computations using customization, code transformations and compilation. Morgan Kaufmann, 2017.
[14] Schaumont, Patrick. "Hardware/software co-design is a starting point in embedded systems architecture education." Proceedings of the Workshop on Embedded Systems Education. 2008.
[15] Practical Introduction to Hardware/Software Codesign By Patrick R. Schaumont book p11-
[16] Book ANT COLONY OPTIMIZATION METHODS AND APPLICATIONS Edited by Avi Ostf eld
[17] Maniezzo, Vittorio, Luca Maria Gambardella, and Fabio De Luigi. "Ant colony optimization." New optimization techniques in engineering 1.5 (2004).
[18] Koudil, Mouloud, et al. "Solving partitioning problem in codesign with ant colonies." Artificial Intelligence and Knowledge Engineering Applications: A Bioinspired Approach: First International Work-Conference on the Interplay Between Natural and Artificial Computation, IWINAC 2005, Las Palmas, Canary Islands, Spain, June 15-18, 2005, Proceedings, Part II 1. Springer Berlin Heidelberg, 2005.

V. Declaration of Originality

I, Jaouaher Belgacem, herewith declare that I have composed the present paper and work by myself and without the use of any other than the cited sources and aids. Sentences or parts of sentences quoted literally are marked as such; other references with regard to the statement and scope are indicated by full details of the publications concerned. The paper and work in the same or similar form have not been submitted to any examination body and have not been published. This paper was not yet, even in part, used in another examination or as a course performance. I agree that my work may be checked by a plagiarism checker.