Ant Colony Optimization

1st Brian Kelein Ngwoh Visas
Electronics Engineering
Hamm-Lippstadt University of Applied Sciences
Lippstadt, Germany
brian-kelein-ngwoh.visas@stud.hshl.de

***Abstract*—This document is a model and instructions for LaTeX. This and the IEEEtran.cls file define the components of your paper [title, text, heads, etc.]. *CRITICAL: Do Not Use Symbols, Special Characters, Footnotes, or Math in Paper Title or Abstract.**

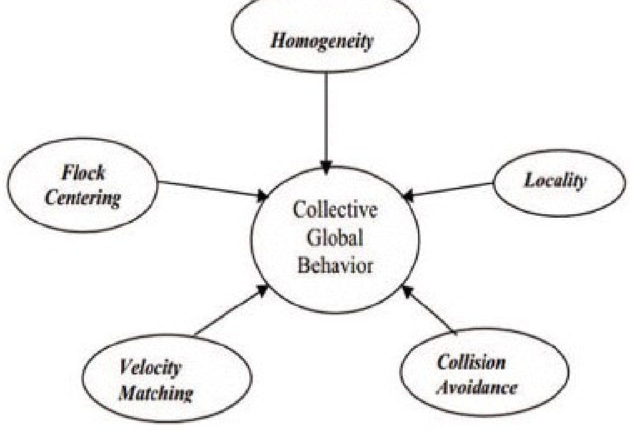
***Index Terms*—component, formatting, style, styling, insert**

I. Introduction

A. Definition of the Algorithm

B. Swarm Intelligence and Biological background

The collective behaviour and social movements of a group of creatures (like bees, birds, fishes, ants, wasps, and many more.) are very significant, but the personal behaviour of a single creature often is uncomplicated. Because of the impressive applicability of the natural swarm systems, researchers have done numerous studies on such behaviours of social creatures. These systems proposed based on Artificial Intelligence (AI) concepts, in the late-80s, by computer scientists. In 1989, the first time, the term "Swarm Intelligence" was applied by G. Beni and J. Wang in the global optimization framework to control the robotic swarm. SI techniques have broad application domains in science and engineering. Therefore, SI is considered a new branch of AI. Hence, modelling the collective behaviour of social swarms in nature like bird flocks, honey bees, ant colonies, and many more. Figure 1 shows an overview of the main properties of collective behaviour.. [1]



[1]

Fig. 1. The main Properties of modelled collective behaviour

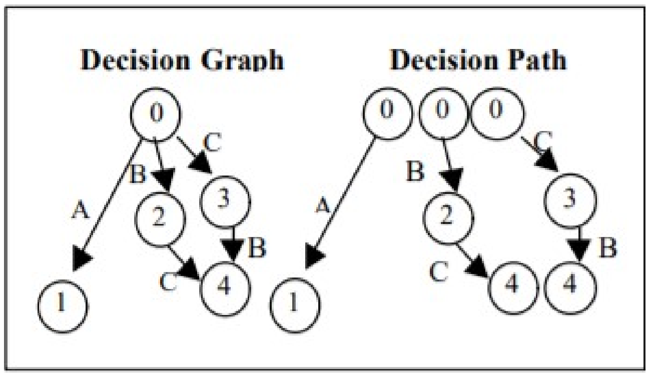
II. Dynamic Ant Algorithm

A. Decision Path

It is a record of the decisions made for a design. Each node of a path represents a design problem state. The first node of the decision path is the starting point of the design process. From the first node, a designer decides to select from all of the possible ways. After that, the designer will reach another state (node) in the design problem. This selection process is done several times until the designer gets a complete design or gets to the dead end.

B. Decision graph

It is a structure that represents a decision space for a design problem. Each node represents a design problem state. Each arc represents a choice made to get to the next node or the next state. It is the same as a decision tree, but in this case, the different sequence of a decision can get to the same state or the same node. Alternatively, each node can have more than one degree. There is a root node, which represents the initial state of the design problem. A path linking the root node to a leaf node is known as a decision path.



[2]

Fig. 2. Decision Tree for Graph and Path

C. Construction of a decision path

It is a process that a sequence of selection is made. This process begins from the first node, which is an initial design state. Then the possible choices are listed, and each possible choice is mapped to a heuristic weight and a pheromone weight as in the Ant algorithm. A choice is selected according to the state transition rule, which is described next, and then the algorithm gets to the new state of the design problem, explained in [2]. This process continues until the design process is completed. While constructing the decision path, the decision graph is also updated.

D. State Transition Rule

It is a rule that will be applied to advance from one design step to the next step by selecting a choice from the possible weighted choice list. If all of the choices were selected (by previous ants), then a choice will be randomly selected by the biases from the pheromone level. This is diverted from another ACO. In ACO, the selection process will be based on a linear combination of heuristic weights and pheromone levels, but in Dynamic Ant, the heuristic weights are used until all choices are selected, then the pheromone levels are considered as seen in [2]. The advantage of our method is that the determinations of heuristic weights are independent of pheromone levels.

E. Path Exploration

It is a process to initialize new paths from the interesting path. The process begins by selecting a random exploration point in the path. Then a path from the root to the exploration point is copied for a new ant as an initial path. Now we introduce the steps of the Dynamic Ant Algorithm. We call this Dynamic Ant because some of the techniques are borrowed from Dynamic Niche Sharing as explained by [2].

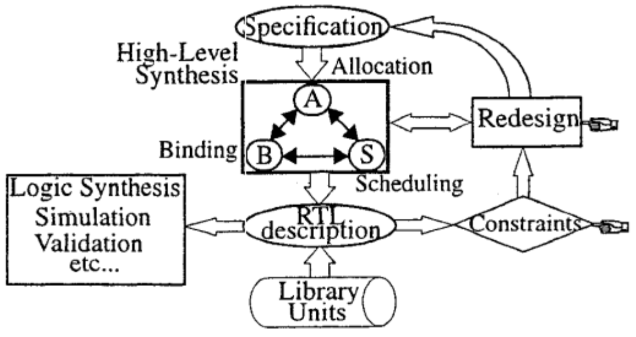
III. High Level Synthesis in Relation to Ant Colony Optimization(ACO

A. High Level Synthesis Flow

HLS Flow defines the process flow from design, right up to the level of synthesis. Here the design is implemented at an abstract level in the form of an algorithm. Furthermore, a tool is used to generate a circuit at Register Transfer Level(RTL) as described by [3], But if the circuit then happens to be in a Hardware Description Language, then it can be simulated and verified.

B. High Level Synthesis Formulation

Let v and w be arithmetic or logical operations and (v, w) be a precedence relation. This means that the output of v must be completed before w can start. Then the problem is choosing hardware h_v, h_w to execute$v, w$and scheduling each into a clock step s so that: $(t_w > t_v + l_V + M)\forall(w, v) \in P$ while minimizing a cost function $C(a, p, d)$ Here t_x is the initiation time of operation $xandl_vh$ is the latency (propagation delay) of operation v, executed on hardware unit h. P is the set of data precedence, and M is a timing margin that accounts for interconnect, mux and register delays. In the cost function, a refers to the total area, p to the power consumption and d to the total latency of the algorithm.



[3]

Fig. 3. HLS Flow

C. High Level Synthesis Methods

1) High Level synthesis Steps: Three related problems are involved in a high-level synthesis: allocation, programming and binding [4]. Attribution means the assignment of sufficient hardware units for all operations. Scheduling is the time allocation for each operation in the detailed behavioural description without any violation of data precedence. Binding means assigning operations at certain times to hardware units. This is the stage in which all architectural interconnections are defined. These tasks are carried out simultaneously with the GA Synthesis (which represents the Ant Colony Optimization). Systems performing these functions in a row may require iteration between steps.

2) High Level Synthesis Constraints: In high-level synthesis, the main constraints of optimization are area and time. Speed (peak and average), throughput (and latency) and clock speed may also include testability, the ability to be routed (usually mux and bus area). The choice of technology, scaling, and design improvements vary in some respects. These constraints To accommodate such changes, a tool of synthesis should be flexible enough.

3) High Level Synthesis Scheduling Methods: The methods of scheduling in the HLS are below. Planning of ASAP/ALAP. As soon as all the necessary information is accessible, ASAP referred to a schedule that assigns operations to be carried out. ALAP refers to the assignment of operations to be performed before their production values are necessary. Many methods are starting from these schedules. Heuristic scheduling, for example, list scheduling [5], orders operations based on priorities and force scheduling [6] that allocate operations based on distributed graphs derived from mobility operations—usually fast but not very flexible heuristic schedulers.Neural Networks scheduling has been implemented using self-organizing algorithms. The disadvantage of this approach is its speed. Simulated annealing scheduling generates random modifications in the schedule and accepts or rejects them according to an arbitrary rule and a parameter that gradually decreases with time. The disadvantage is the speed. Integer Linear Programming scheduling (ILP) is an exact scheduling technique. ILP scheduling is formulated as an optimization problem using certain constraints [3]. This method suffers from increasing time-complexity since it is exponential in the worst case. This puts more significant, more realistic problems out of their scope.

D. *Genetic Algorithm Setup*

The use of GA parameters is Type z of population: The complete circuit is a chromosome. There are 100,000 to several thousand typical population size for synthesis problems. The number of generations g maximum: This is the typical problem of tens of thousands of generations. In many cases, however, within one hundred, a good industrial solution is found. Typically, the crossover probability is between 20 and 60 per cent. Unless otherwise noted, the solutions shown in [3] use a 50 per cent overlap probability. Typically, the probability of mutation is between 0.1 and 1 per cent. It was set to 0,5per cent, for example, in this paper. The first circuits were generated randomly. For most examples shown in [3], a single point crossover was used. The method of selection was the roulette wheel that was replaced with a steady state. The halt criteria were the number of generations.

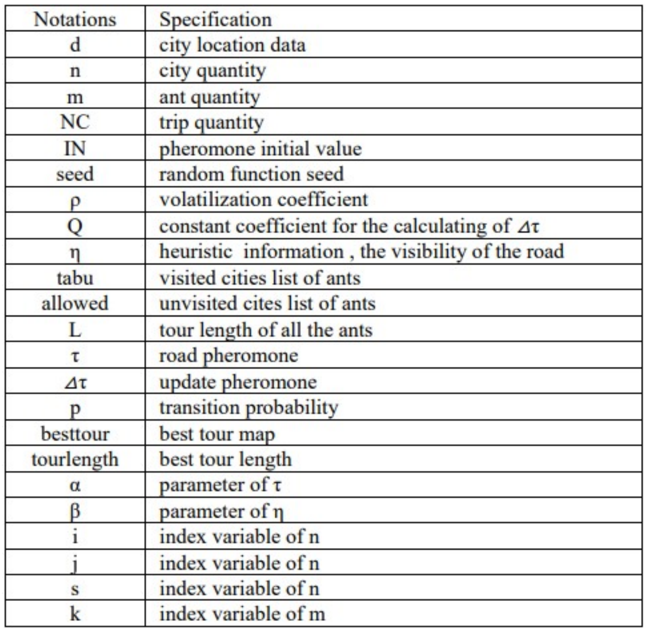
E. *Chromosome encoding*

The chromosome comprises two fields per operation (hw, cc), where hw is the field of the id of the FU used, and cc is the relative cycle for the start of the mobility-based operation. Mobility is from the start of the operation (ASAP) to the latest time interval, where an operation can be executed (ALAP). It is no longer than the critical path that ALAP allows.For small increases in time, minimal hardware solutions can often be found. Therefore, as resource-constrained as possible, we define and use an ARAP program to schedule the minimum possible hardware. The achievement of such a schedule has the same complexity as programming. Furthermore, this schedule can be relaxed to encode, and the upper limits can be used rather than the optimum tasks so that all reasonable solutions are represented in the design space.The number of bits required to represent a CDFG is given by: $n = \sum_{v=1}^{v} log_2 K_v + mob_v$. Where K is the number of units available that can perform mobility v. mob and V, the number of operations is total. The order of the operations in the chromosome is chosen to precede all of the operations in the graph. The objective function calculation thus reduces the complexity.

IV. HACO-F: An Accelerating High Level Synthesis Based Floating Ant Colony Optimization on FPGA

A. *HACO-F Design based on High Level Synthesis*

HACO-F is designed on HLS by integrating the Ant System (AS) algorithm, as the computation scale is more prominent than those of other ACOs. The HACO-F design is composed of two aspects. One is data optimization design, which can redefine the precision of variables for further reduction of resource usage. The other is loop optimization design, which makes the AS algorithm paralleled and integrated into an FPGA. The notations of HACO-F are explained below [7]



Notations	Specification
d	city location data
n	city quantity
m	ant quantity
NC	trip quantity
IN	pheromone initial value
seed	random function seed
ρ	volatilization coefficient
Q	constant coefficient for the calculating of Δτ
η	heuristic information , the visibility of the road
tabu	visited cities list of ants
allowed	unvisited cites list of ants
L	tour length of all the ants
τ	road pheromone
Δτ	update pheromone
p	transition probability
besttour	best tour map
tourlength	best tour length
α	parameter of τ
β	parameter of η
i	index variable of n
j	index variable of n
s	index variable of n
k	index variable of m

[7]

Fig. 4. Notations of HACO-F

B. *HACO-F Data Optimization Design*

Data optimization is to redefine the precision of the variables used in the algorithm to reduce resource usage further. The process of data optimization mainly focuses on the boundaries of input variables, which may differ in different applications. According to the boundaries of input variables, it is easy to redefine the valuable bits of these input variables. Especially for float type variables, we redefine the precision by the following transition principle. For the precision insensitive variables, 6bits in decimal (20bits in binary) is defined, and the total bits should be defined smaller than 32bits (the same with float type). Otherwise, there would be little optimization or even worse. For some precision sensitive variables, the total bits may be up to 64bits (the same with double type). In particular, we may approximate the percentage values with 6bits in binary. Other variables, valuable bits, are defined based on the relations, such as the operations relations and valid bits dependency relations.

C. *HACO-F Loop Optimization*

V. Ant Colony Optimization Algorithm For Fuzzy Controller And Design Of Its FPGA

A. *Basic Concepts Of Fuzzy Controller and Ant Colony Optimization*

a) Fuzzy Controller: The Ith Rule Let i be an index for a fuzzy rule number, The ith rule denotes:
R_i : If $x_1(k)$ is A_{i1} And ..., And $x_n(k)$ is A_{in}
Then $u(k)$ is $a_i(k)$ (1)

Also we have μ_A which is referred to as the membership degree for fuzzy set A. Asembling all these input together and considering that the fuzzy system consists of s fuzzy rules, the Weighted Average Deffuzzification method system output is computed by the equation below

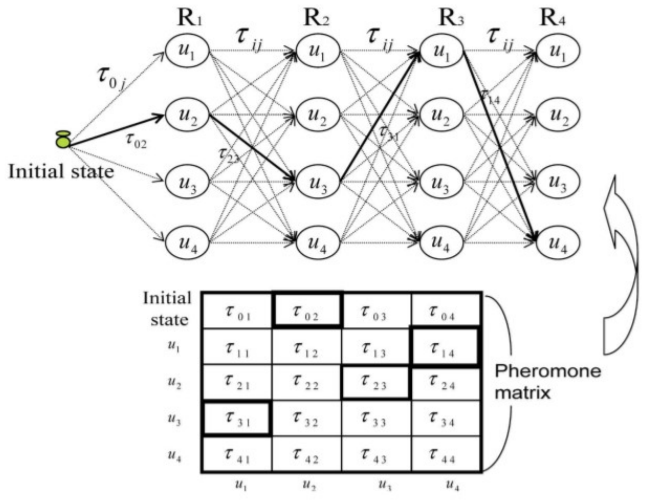
$u = \frac{\sum_{i=1}^{r} \phi_i(\overrightarrow{x}) a_i}{\sum_{i=1}^{r} \phi_i(\overrightarrow{x})}$.

Here k is the time step, $x(k)$ is the input variable, $u(k)$ is the output action variable, A is a Fuzzy set, $a_i(k)$ is a recommended control action which is a fuzzy singleton. As analysed by [8], the inference engine, the fuzzy and Operation are implemented by the algebraic minimum of the fuzzy theory. $\overrightarrow{x} = (x_1, \ldots, x_n)$, is an input data set.There is also what we call the firing strength denoted by $\phi_i(\overrightarrow{x})$

b) Ant Colony Optimization(ACO): The ACO meta-heuristic was inspired by actual ant behaviour, particularly on how they forage for food. With these algorithms, a colony of artificial ants with a finite size is created, and each ant develops a solution. The performance measure is based on a quality function F(). ACO algorithms can be applied to problems described by a graph consisting of nodes and edges connecting the nodes. Optimization problem solutions can be expressed in terms of feasible paths on the graph. Among the feasible paths, ACO algorithms attempt to find the one with minimum cost. The problem of selecting fuzzy rule consequences can be described and solved by ACO algorithms. The data collected by ants are stored in the pheromone trails in association with edge links during the search process. The ants work together to find a solution by exchanging data through the pheromone trails. Edges also have the associated heuristic value , a preliminary piece of information on the definition of the problem instance or the run-times of information from a source other than ants. Ants can act simultaneously and independently, demonstrating cooperation.

B. Fuzzy Controller Design By Ant Colony Optimization

Here we look at how the ACO is directly applied to the Fuzzy controller system. First we have the predecessor part which is partitioned well ahead of time with relative ease, then we have input variable, and the sum of fuzzy set combinations of this input variables.This sum corresponds to the fuzzy rules in the product space. Also we have to select the best candidate action from a total of fuzzy controller combinations together with each r fuzzy rule. This shows us that there is a combinational optimization problem all this is properly analysed in [8]. This leads to a problem of combination optimisation. Because the ACO algorithm is proposed using the ant population to solve discrete problems in combination, this motivates the ACO proposal for the design of FC parts known as ACO-FC. ACO-novelty FC's is the proposed definition of a new pheromone matrix and heuristic values for FC design. This means that the research on an entirely new ACO algorithm in ACO-FC is not in focus. Trip determination from the current fuzzy rule to the following fuzzy rule is based on pheromone trails. Searching for the best one among all fuzzy rule combinations is pheromone level based. In the ACO-FC system, a colony of N_a ants is generated and placed in the nest. The terminal condition in this paper is the total number of search iterations performed by the ant colony. According to the pheromone level, the ant moves through a trip that corresponds to the choice of all fuzzy rule consequent actions. The pheromone level is stored in a matrix shown in Fig. 6. The special value τ_{0j} determines the control action selection of R1 when an ant moves out from its nest. It is known that control voltage increase causes a higher temperature. For the candidate control actions in each fuzzy rule, two heuristic values are assigned to two sets of control actions: positive and negative.As [8] would go further to explain, determination of the heuristic value usually requires a priori information about the problem instance. The controlled plant model is assumed unknown, except for the change of output direction with a control input. For example, in a temperature control system, it is known that the increase in control voltage causes the temperature to rise.

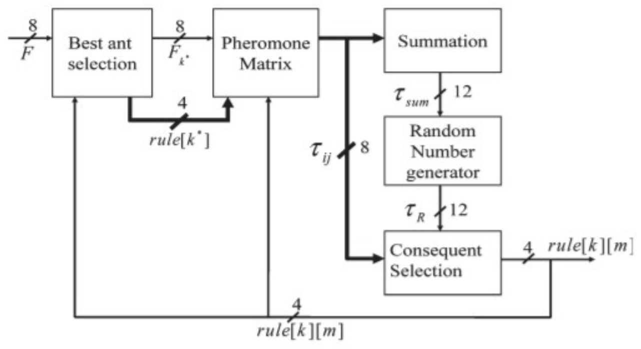

R1
R2
R3
R4
τ_{ij}
τ_{0j}
τ_{02}
τ_{22}
τ_{31}
τ_{14}
Initial state
u_1
u_2
u_3
u_4
Initial state
τ_{01} τ_{02} τ_{03} τ_{04}
τ_{11} τ_{12} τ_{13} τ_{14}
τ_{21} τ_{22} τ_{23} τ_{24}
τ_{31} τ_{32} τ_{33} τ_{34}
τ_{41} τ_{42} τ_{43} τ_{44}
Pheromone matrix

[8]

Fig. 5. FC constructed by an Ant trip and the corresponding pheromone matrix

C. Hardware Implementation Of Ant Colony Optimization in ACO-FC

The ACO algorithm in the proposed ACO-FC is hardware implemented by FPGA. A PC simulates the designed FC and the controlled plant to test the design ACO chip performance. Configuration is considered online control. Like a practical plant online control, performance evaluation data are available only when the control starts. That is, at a time, only one Ant can be evaluated. Thus, the selection and evaluation functions of each Ant in the proposed ACO chip are implemented sequentially. The input from PC to FPGA is the signal F, denoting performance measure and representing eight bits. The FPGA output is the signal rule$[k]$ $[m]$, which sends information on the selected consequent action for fuzzy rule m by the kth Ant. The architecture of FPGA-implemented ACO is shown in Fig. 2. It consists of five modules. The summation, random number generator, and consequent selection modules select the FC and activate each new ant path-building process. The best ant selection module is activated for each new ant path-building process, but it works after controller evaluation in PC. Pheromone matrix update is performed in the pheromone matrix module and is activated only once in each iteration. That is, it is activated after all ants have built their paths. For illustration, assume that the candidate action number N is 16. The signal rule[k][m] is represented by four bits representing one of the 16 candidate actions. Detailed descriptions of the five modules are as follows.



[8]

Fig. 6. Architecture of Hardware-Implemented ACO

VI. Conclusion

References

[1] A. Mohammadi and S. H. Zahiri, "Analysis of swarm intelligence and evolutionary computation techniques in iir digital filters design," in *2016 1st Conference on Swarm Intelligence and Evolutionary Computation (CSIEC)*, 2016, pp. 64–69.

[2] R. Keinprasit and P. Chongstitvatana, "High-level synthesis by dynamic ant," *International journal of intelligent systems*, vol. 19, no. 1-2, pp. 25–38, 2004.

[3] E. Torbey and J. 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)*, 1998, pp. 224–229.

[4] G. De Micheli, *Synthesis and optimization of digital circuits*. McGraw Hill, 1994, no. BOOK.

[5] R. A. Walker and R. Camposano, *A survey of high-level synthesis systems*. Springer Science & Business Media, 2012, vol. 135.

[6] P. G. Paulin and J. P. Knight, "Force-directed scheduling for the behavioral synthesis of asics," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 8, no. 6, pp. 661–679, 1989.

[7] S. Zhang, Z. Huang, W. Wang, R. Tian, and J. He, "Haco-f: An accelerating hls-based floating-point ant colony optimization algorithm on fpga." *International Journal of Performability Engineering*, vol. 13, no. 6, 2017.

[8] C.-F. Juang, C.-M. Lu, C. Lo, and C.-Y. Wang, "Ant colony optimization algorithm for fuzzy controller design and its fpga implementation," *IEEE Transactions on Industrial Electronics*, vol. 55, no. 3, pp. 1453–1462, 2008.