Expert Systems with Applications 38 (2011) 3632–3639

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

High speed ant colony optimization CMOS chip

Kazem Gheysari *, Abdollah Khoei, Behboud Mashoufi

Microelectronics Research Laboratory, Urmia university, Urmia, Iran

ARTICLE INFO

Keywords:
Ant colony optimization
TSP problem
CMOS implementation
Pheromone matrix

ABSTRACT

Ant colony optimization (ACO) is an optimization computation inspired by the study of the ant colonies' behavior. This paper presents design and CMOS implementation of the ant colony optimization based algorithm for solving the TSP problem. In order to implement ant colony optimization algorithm in CMOS, we will present a new algorithm. This algorithm is based on the original ant colony optimization but it can be implemented in CMOS. Briefly, pheromone matrix is transformed on the chip area and ants move up-down through the pheromone matrix and they make their decisions. Finally ants select a global path. In previous researches only pheromone values is used, but select the next city in this paper is based on heuristics value and pheromone value. In definition of problem, we use heuristics value as a matrix. Previous researches could not be used for wide type of optimization problem but our chip gives heuristics value initially and we can change initial value of heuristics value according to the optimization problem so this capability increases the flexibility of ACO chip. Simple circuit is used in blocks of our chip to increase the speed of convergence of ACO chip. We use Linear Feedback Shift Register (LSFR) circuit for random number generator in ACO chip. ACO chip has capability of solving the big TSP problem. ACO chip is simulated by HSPICE software and simulation results show the good performance of final chip.

1. Introduction

From viewpoint of software, there are plenty of researcher being done on information technology, particularly in artificial intelligence (AI). In contrast, there has been little study on hardware because of the difficulty of producing and using general-purpose versions, therefore we are attempting to realize AI using CMOS technology.

Natural evolution has yielded biological systems in which complex collective behavior emerges from the local interaction of simple components. One example where this phenomenon can be observed is the foraging behavior of ant colonies. In ACO, a number of artificial ants build solutions to the considered optimization problem at hand and exchange information on the quality of these solutions via a communication scheme that is reminiscent of the one adopted by real ants. Different ant colony optimization algorithms have been proposed. The original ant colony optimization algorithm is known as Ant System (Dorigo, Maniezzo, & Colorni, 1991, 1996) and was proposed in the early nineties. Ant colony optimization has been formalized into a Metaheuristics for combinatorial optimization problems by Dorigo and Di Caro (1999), Dorigo, Di Caro, and Gambardella (1999).

Computational technique implementations, such as neural networks (Jung & Kim, 2007), fuzzy controllers (Juang & Chen, 2006; Li, Chang, & Chen, 2003), adaptive fuzzy controller (Juang & Hsu, 2005), and GA (Aporntewan & Chongstitvatana, 2001), Field-Programmable Gate Arrays (FPGA) chip have been proposed. In Merkle and Middendorf (2002), they studied the problem of implementing the ACO on large processor arrays with small processors that have only a constant number of registers. The main new features of their work were the non-generational approach and the use of a threshold based decision function for the ants. This was first implementation of the ACO approach on a reconfigurable architecture.

In Scheuermann et al. (2004), a population-based ACO (P-ACO) is implemented on an FPGA chip and the designed chip is applied to a Single Machine Total Tardiness Problem (SMTTP). They demonstrated that a straightforward hardware mapping of the standard ACO algorithm is not very well suited to implementation on the resources provided by current commercial FPGA architectures. Instead, they suggested using the Population-based ACO (P-ACO), in which pheromone information is replaced by a small set (population) of good solutions discovered during the preceding iterations. Accordingly, the combination of pheromone updates and evaporation has been replaced by inserting a new good solution into the population, replacing the oldest solution from the population. Experimental results indicated that P-ACO performs at least as well as the standard ACO approach (Scheuermann et al., 2004). In Nakao and Izumi (2004), an ant chip is designed and fabricated. They realized hardware based on the foraging behavioral model using a hardware description language (HDL). In this work, they by extracting ant foraging behavior in detail, integrating the

* Corresponding author.
E-mail address: kgheysari@gmail.com (K. Gheysari).

0957-4174/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2010.09.017

function onto one chip, and constructing the hardware environment on an LSI. They fabricated AI-LSIs on a chip for the first time using an ant model. Their designed chip had 111 pins for quasi-TSP problem with three cities and one nest, therefore the number of required pins for real world application with many cities is very much so their approach is not applicable. In Juang, Lu, Lo, and Wang (2008), ACO applied to Fuzzy Controller (FC) design, called ACO-FC, for improving design efficiency and control performance, as well as ACO hardware implementation. The ACO used in ACO-FC is based on the known ant colony system and is hardware-implemented on a FPGA (Field Programmable Gate Array) chip. They had used external PC to calculate performance measure per ant in current iteration but our proposed chip, itself calculates performance measure of ant and stores it in the solution memory. It uses solution memory at end of iteration to find the best ant in the iteration.

In this paper, CMOS implementation of Ant Colony Optimization algorithm is presented. Designed chip takes heuristics values and pheromone values initially, and then it starts its operation. We can change the heuristics value according to the problem. We have used simple circuits in our design to increase the speed of operation.

Final chip is tested by TSP problem but it can be used for another optimization problem in which we define the problem by heuristics values (heuristics matrix). Selection scheme for next city is based on Roulette wheel in GA (Genetic Algorithm). It causes that our ACO chip is converged in less iteration than previous approaches. Therefore, our chip finds the optimum path very earlier than previous approaches.

The remainder of this paper is organized as follows. Section 2 provides a brief introduction of Ant Colony Optimization (ACO) algorithm. In Section 3, the proposed ACO algorithm is presented and CMOS implementation of the proposed ACO is introduced in Section 4. Timing of pulse of ACO chip is described in Section 5. Simulation results are shown in Section 6. Finally, Conclusion is presented in Section 7.

2. Ant colony optimization

Here, we briefly introduce ACO and its application by using TSP as an example (Dorigo, Birattari, & Stützle, 2006; Dorigo & Di Caro, 1999; Dorigo et al., 1991, 1996, 1999). In the TSP, a given set of n cities has to be visited exactly once and the tour ends in the initial city. We denote the edge between city i and j as (i,j) and its distance as $d_{ij}(i,j \in [1,\mathrm{n}])$. Let $\tau_{ij}(t)$ be the intensity of pheromone on (i,j) at time t, and use $\tau_{ij}(t)$ to simulate the pheromone of the real ants. Suppose m is the total number of ants in each iteration and ant k in city i selects next city j according to the following probability distribution:

$$p_{ij}(t) = \begin{cases} \frac{\tau_{ij}^{\alpha}(t)\times\eta_{ij}^{\beta}(t)}{\sum_{r\in allowed_k}\tau_{ir}^{\alpha}(t)\times\eta_{ir}^{\beta}(t)} & j \in allowed_k \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

where $allowed_k$ is a set of the cities can be chosen by the kth ant at city i for the next step, η_{ij} is a heuristic function which is defined as the visibility of the link between cities i and j. For TSP problem it can defined as $1/d_{ij}$.

The relative influence of the trail information τ_{ij} and the visibility η_{ij} are determined by the parameters α and β. The intensity of pheromone is updated by the Eq. (2)

$$\tau_{ij}(t+1) = \rho\tau_{ij}(t) + \Delta\tau_{ij} \quad (2)$$

where $0 < \rho < 1$ represents the evaporation of $\tau_{ij}(t)$ between time t and $t+1$, $\Delta\tau_{ij}$ is the increment of the pheromone on (i,j) in step t. Eq. (3) shows the formula of $\Delta\tau_{ij}$ based on $\Delta\tau_{ij}^k$. $\Delta\tau_{ij}^k$ is the pheromone laid by the kth ant on it. It takes different formula depending on the model used.

$$\Delta\tau_{ij} = \sum_{k=1}^{m}\Delta\tau_{ij}^k \quad (3)$$

For example, in the most popularly used model called "ant circle system", it is given as Eq. (4).

$$\Delta\tau_{ij}^k = \begin{cases} \frac{Q}{L_k} & \text{if the } k\text{th ant passes(i, j)in current tour} \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

where Q is a constant and L_k is the total length of current tour traveled by the kth ant.

In hardware realization of ACO, the Eq. (1) often change (Scheuermann et al., 2004; Nakao & Izumi, 2004). For example, the heuristic information (η_{ij}) usually ignore for generality and to ease the implementation so we do not need multiplier and divider circuit. In our work, we present a new scheme for selection of next city based on Rollout wheel and we do not ignore heuristic (η_{ij}). In most work, α is one to avoid exponentiation because exponentiation circuit needs very large space on the chip area and it needs high power consumption.

3. Proposed ACO algorithm

When we want to implement an artificial intelligent algorithm, there are two options. First is software implementation using personal computer and second is hardware implementation. If computer be used, we can implement the original algorithm and there is not any limitation for software implementation. Conditional commands or loops in the program can be used and we can define our variables with desirable bits but a designer of integrated circuit cannot implement the original algorithm on the chip area because of some limitation such as register length, speed, power consumption, delay and jitter (Razavi, 2000; Weste & Eshraghian, 1994), so the original algorithm must be changed. These changes are one of the most important things in hardware implementation of artificial intelligence algorithm. These changes should not decrease performance of the final algorithm (Gheysari, Khoei, & Mashoufi, 2009).

The proposed ACO is very similar to original ACO. This algorithm is based on original ant colony optimization but it can be implemented in CMOS. The general principle of proposed ACO algorithm is based on the algorithm used in Scheuermann (2005). The principle of this algorithm is to embed the $n \times n$ pheromone matrix M into an $n \times n$ cell so that $cell_{ij}$ contains only the pheromone value τ_{ij}, i, $j \in [1{:}n]$. The ants are then pipelined through the pheromone matrix. Flowchart of the proposed ACO algorithm is shown in the Fig. 1. Fig. 2 shows the movement of ant through the pheromone matrix.

First stage is RESET and second stage is load of initial values of cells (heuristics & initial pheromone) according to the optimization problem. Then, first iteration starts and ant counter increases one unit and first ant ($a = 1$) begins its tour through pheromone matrix. One of the cells in row 1 is selected by ant 1. Suppose, ant 1 selects cell 2 in row 1. At first, all elements of selection set of ant 1 are one, but when ant 1 selects a cell in row 1, the corresponding item in selection set ($S_{a1,2}$) becomes zero. Therefore, ant 1 does not select cell 2 in other rows of the chip and it prevents cycle in the algorithm. Connection of cells to prevent cycle in the algorithm is very simple, and Fig. 3 shows this connection for 16 cells. Finally, at the final row of chip, all elements in selection set become zero and the ant has made a closed tour.

The pheromone value of the selected cell decreases (local pheromone update in ACO).

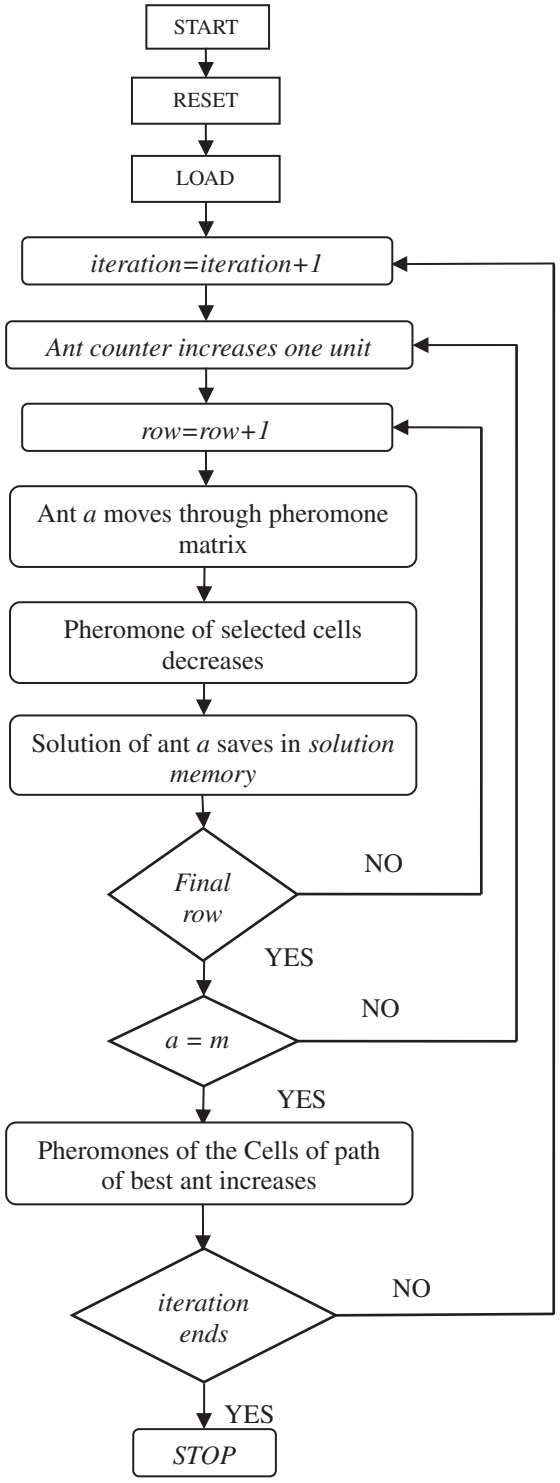

Fig. 1. Flowchart of proposed ACO algorithm.

Pheromone is implemented by a loadable decremental counter. Thereby, reducing the attractiveness of the respective arc for following ants. Consequently, the exploration of not yet visited arcs is supported and ants are less likely to converge to a common path in the chip.

Every cell has a memory in itself. When an ant selects the cell, this memory saves selection of ant with index of ant. At the end of iteration, the path of best ant will be found by reading this memory in each cell. Finally, we find the path of the best ant with this memory in each cell, in others words, this memory shows that if ant i had selected this cell in its path.

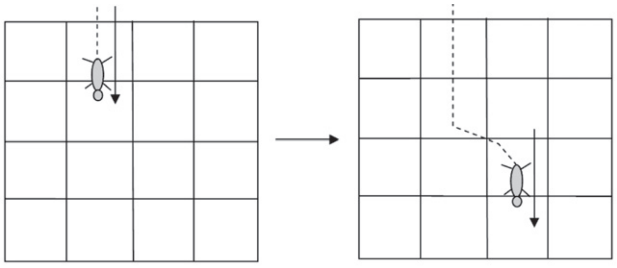

Fig. 2. Movement of ants through pheromone matrix.

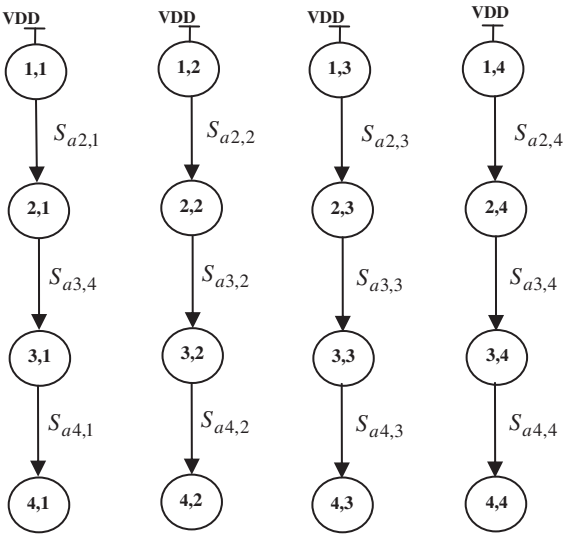

Fig. 3. Connection of cells in ACO chip for 16 cells.

A circuit as Best Ant Finder is designed to find the best ant. The operation of Best Ant Finder will be described in next section. The output of the best ant finder circuit is the index of the best ant. The best ant have maximum sum of heuristics at the end of iteration. After finding the cells of path of best ant, the pheromone value of these cells is added with a constant (Global pheromone update in original ACO). Thus, some amount Δ of pheromone is added to element τ_{ij} of these cells of ACO chip (pheromone circuit of cells). This procedure executes a number of iterations until a specified stopping criterion has been met, e.g. a predefined maximum number of iterations have been executed, a specific level of solution quality has been reached, or the best solution has not changed over a certain number of iterations.

3.1. Selection of next city

Next cell is selected by Next City Selector Circuit. This circuit calculates the following equation for ant a with selection set (S_a) in row $i-1$:

$$p_{ij} = \sum_{k=1}^{n} S_{aik}\tau_{ik}\eta_{ik} \tag{5}$$

Then a random number generator circuit generates a random number (r_i) and if $p_{i,j-1} \leqslant r_i < p_{ij}$ then $cell(i,j)$ will be selected and item j in selection set gets zero.

3.2. Best ant

The property for selection of the best ant at the end of iteration is sum of heuristics of cells which is selected by the ant in its path through ACO chip. Eq. (6) shows it:

$$p_{ij} = \sum_{k=1}^{n} \eta_{ik} \tag{6}$$

At the end of the iteration, the ant that has maximum sum of heuristics is best ant in that iteration. This property, for the first time, is used for selection of best ant. In TSP problem, heuristics value is inverse of distance between cities. Inverse of distance must be multiplied with a constant. Result is rounded to nearest integer, and then this integer loads into cells of ACO chip. The bit length which is used for heuristics is 8 then heuristics value must be lower than 256.

4. CMOS implementation of proposed ACO

In this section, the proposed algorithm is implemented in CMOS technology and different blocks of ACO chip are presented. Fig. 4 shows the architecture of ACO chip. This chip contains 9 cells. This small architecture is selected for better description but it can be used for bigger problem with more cells.

ACO chip contains many block but we describe the most important blocks in the following sections.

4.1. Cell

Cell is the main block in ACO chip and it has many tasks. Fig. 5 shows the block diagram of the cell. The timing diagram of cell will

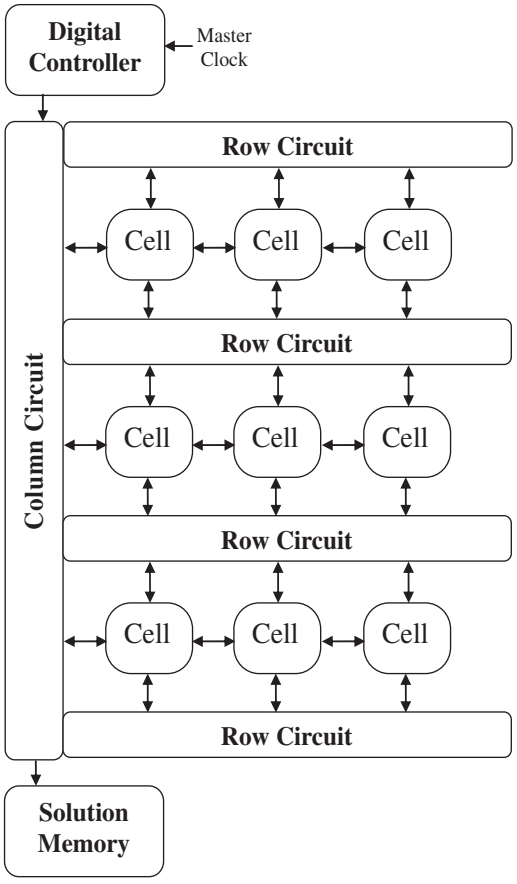

Fig. 4. Architecture of proposed ACO chip.

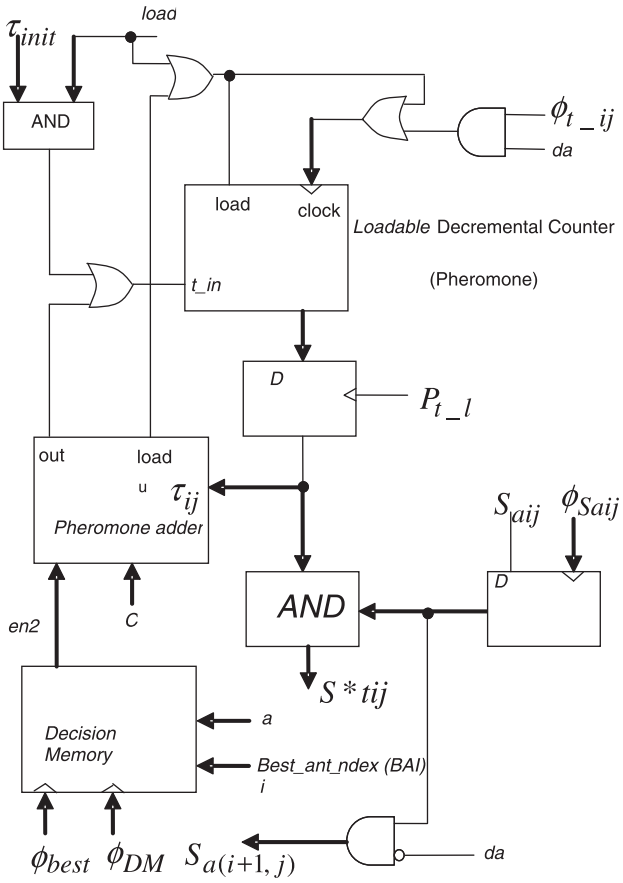

Fig. 5. Block diagram of Cell.

be described in the Section 5. Most important blocks of Cell are: a loadable decremental counter, decision memory, and pheromone adder. Loadable decremental counter has role of pheromone in the cells of ACO chip. It takes initial value of pheromone and starts its work. Every ant is allowed to update and evaporate a certain amount of pheromone directly after is has found a solution. This amount of evaporation is one unit.

Decision memory saves the decision of ant with index of ant. Reading decision memory shows that if ant *i* had selected this cell in its path. Fig. 6 shows the schematic of Decision Memory.

At end of iteration, for finding the cells of path of best ant, pheromone adder block adds a constant with current value of pheromone.

4.2. Next city selector circuit

This circuit selects next city based on pheromone and heuristics values of cells in the next row of chip. It works randomly, but probability of select a cell with more pheromone value and more heuristics value is high. In other words, every selectable item has at least some small probability of being chosen by an ant.

There is a random number generator circuit in each row of cell that generates a random number. Random number comes to the next city selector circuit as *ri*. Block diagram of next city finder circuit for four cities is shown in the Fig. 7. It is for 8 bit but for big problem with many cities, we can increase the number of bit easily because in bigger problem, $\sum_i \eta_{ik}$ cannot be presented by 8 bit and we need bigger bit register width.

Next City Selector Circuit consists of four stages. First block is digital multiplier that multiplies n_i with $S * tij_i$. Second block is

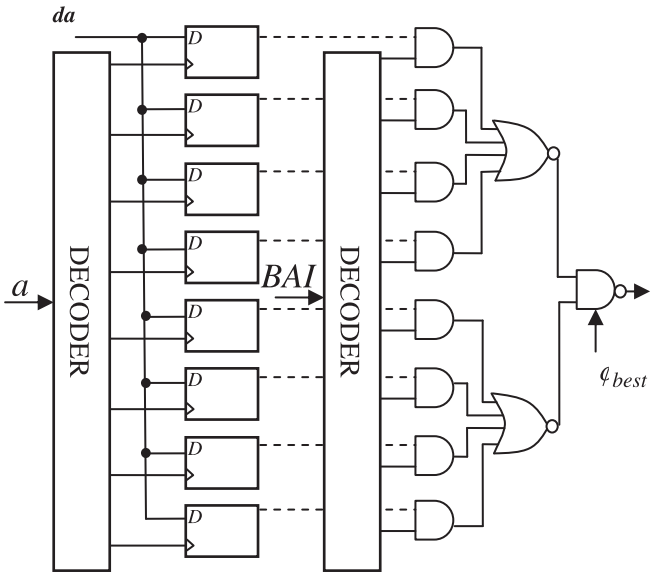

Fig. 6. Schematic of solution memory.

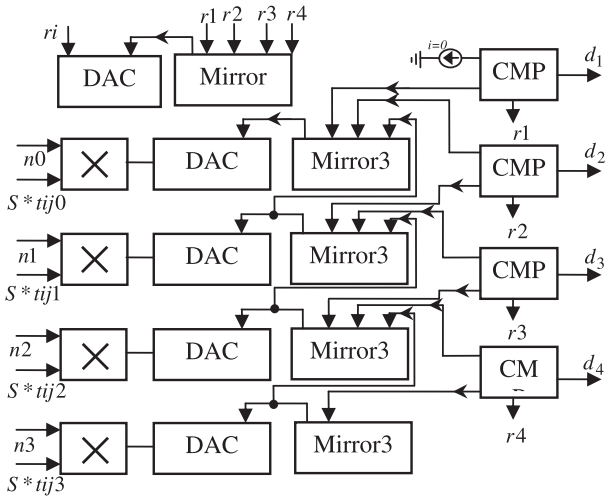

Fig. 7. Next city selector circuit.

DAC. Output of multiplier is connected to DAC. DAC behaves similar to Current Mode DAC and converts digital data from multiplier to analog data (current). Schematic of DAC is shown in the Fig. 8.

After DAC is mirror3. It mirrors input current in output branches. Output current of mirror3 goes to CMP block. If $I_{in1} < I_{ri} < I_{in2}$ then output becomes high (logical one = 3.3 V).

4.3. Comparator circuit

Comparator circuit is shown in the Fig. 9. Inputs of Comparator are currents, in others words, it is a Current-Mode analog comparator. Output is digital and connects to the corresponding cell of ACO chip, which is shown in the Fig. 7. If $I_{in1} < I_{ri} < I_{in2}$ then output becomes high (3.3 V).

4.4. Solution memory

There is a solution memory in ACO chip. You can see the place of solution memory in ACO chip in the Fig. 4. Solution memory finds

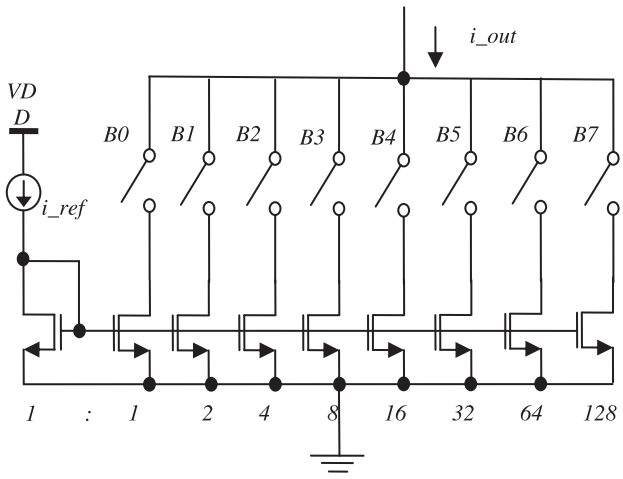

Fig. 8. Schematic of DAC of next city selector circuit.

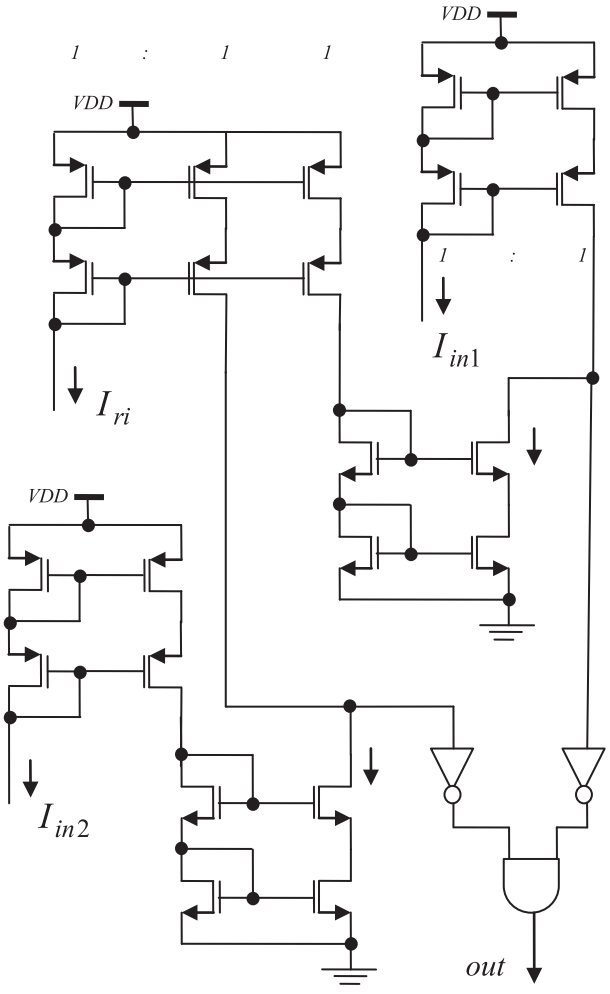

Fig. 9. Comparator circuit.

the index of best ant in current iteration. It compares the sum of heuristics of each ant to last ant and if sum of heuristics of current ant is bigger than last ant then it saves the index of current ant as best ant index. Schematic of Solution memory is shown in the Fig. 10.

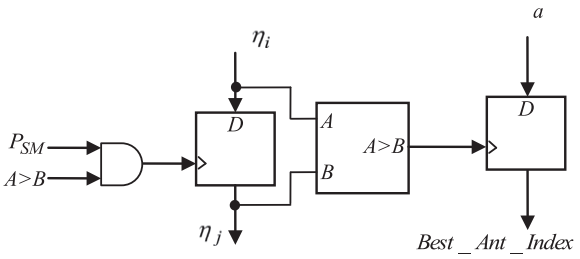

Fig. 10. Solution memory circuit.

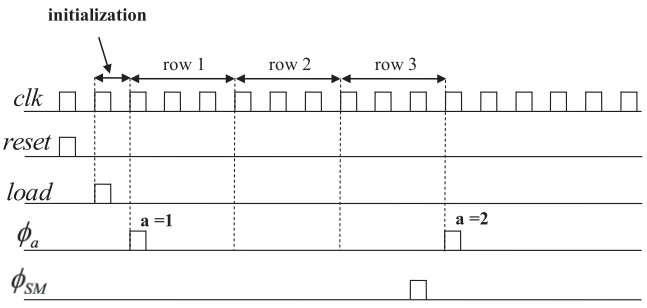

Fig. 12. Timing of five signals (clock, reset, load, ϕ_a, ϕ_{SM}).

4.5. Random number generator

There is a random number generator in each row of ACO chip. This block generates a random number for row *i*. RNG works based of LSFR (Linear Feedback Shift Register).

A linear feedback shift register (LFSR) is a shift register whose input bit is a linear function of its previous state. LSFR is shown in the Fig. 11 which has 15 bits. Polynomial of LSFR is:

$$p(x) = x^{13} + x^9 + x^8 + x^7 + x^5 + x^0 \quad (7)$$

5. Timing

Timing of ACO chip pulses is described in this section. We have designed a digital controller to generate the control signals for ACO chip. Five important pulses are shown in the Fig. 12. At first, reset = 1 and then initial values is loaded (load = 1). Now, first ant ($a = 1$) starts its tour and ant counter increases one unit. This ant moves through chip, when this ant gets final row, its solution saves in the Solution Memory ($\phi_{SM} = 1$). This process repeats for all ants. In this figure, we supposed that three is three rows in ACO chip.

After making solution by each ant, ϕ_{best} becomes high and index of best ant in current iteration is found. This timing is shown in the Fig. 13. In this figure, we supposed there are three ants in each iteration.

Every cell has specific timing which is shown in the Fig. 14. This timing does not have any relation to number of cell in ACO chip. It has three stages. At first stage, initial values is loaded and then random number is read from Random Number Generator ($\phi_{ri} = 1$). Afterwards, next city selector selects a cell in row randomly, and this cell saves in Decision Memory of the selected cell.

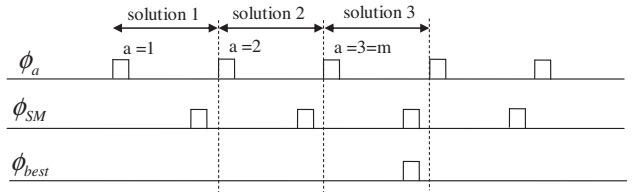

Fig. 13. Timing of three signal (ϕ_a, ϕ_{SM}, ϕ_{best}).

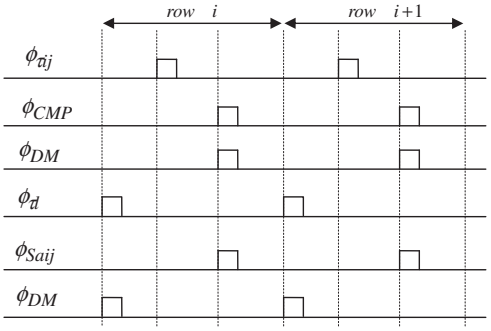

Fig. 14. Timing of signals of CELL.

6. Simulation results

Several simulation cases were conducted to assess the performance of the ACO chip. In this section, the simulation results of ACO chip for different TSP problem will be presented. Simulation results are for HSPICE software in 49 level 0.35 μm CMOS technology. Input clock is 50 MHz. The parameters used for the test runs are: $m = 8$, $\alpha = 1$, $\beta = 1$, $C = 7$.

We tested the optimization behavior of ACO chip on TSP instances TSP-LIB (Reinelt, 1991).

Fig. 15 shows the final answer of ACO chip for TSP problem with 10 cities. This ACO chip contains 100 cells. It can be seen from the Fig. 15 that ACO chip approximately is converged after 400 iterations (8 μs). Then, ants have selected a common path in ACO chip and there is not any change in sum of heuristics of best ant.

Fig. 16 is answer of ACO chip for TSP problem with 17 cities. It is converged in 1200 iterations (24 μs). It is understood form this experiment that ACO chip needs more time for big problem.

Sum of heuristics of best ant in ACO chip with 48 cells (2304 cells) is shown in the Fig. 17. It can be seen from the Fig. 17 that ACO chip is converged in 35,000 iterations (0.7 ms).

We used many integrated circuit design techniques in ACO chip to increase the capability of final chip in solving big TSP problem

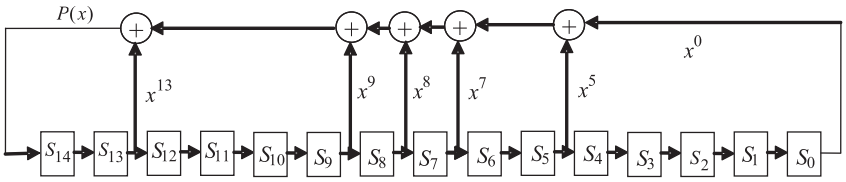

Fig. 11. Linear Feedback Shift Register (LSFR) circuit.

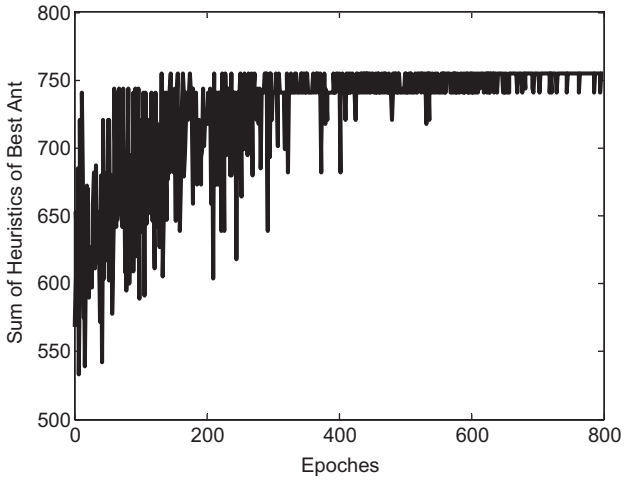

Fig. 15. Sum of heuristics of best ant in ACO chip with 10 cities (100 cells).

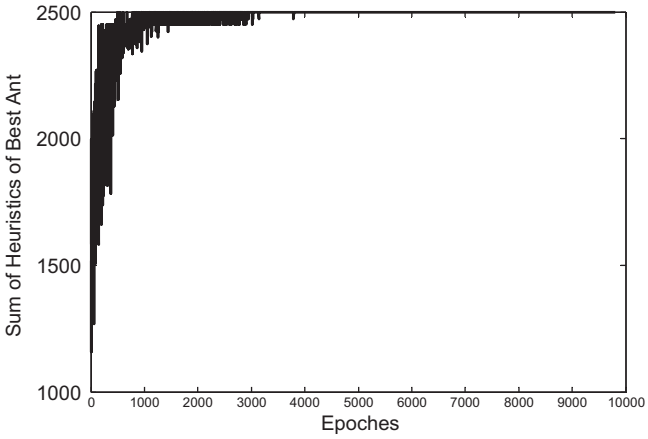

Fig. 16. Sum of heuristics of best ant in ACO chip with 17 cities (289 cells).

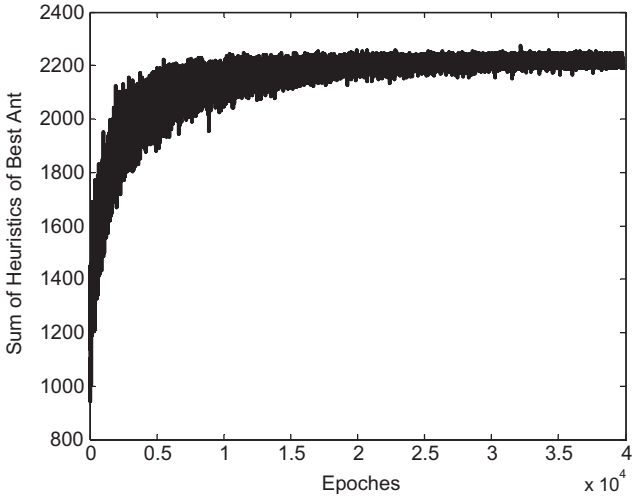

Fig. 17. Sum of heuristics of best ant in ACO chip with 48 cities (2304 cells).

but proposed ACO chip cannot be used for bigger TSP problem than TSP problem with 48 cities.

There are many reasons. Clock signals are the heartbeats of digital systems. Capacitor of some nodes in the chip becomes very much and we cannot drive this large capacitor. To solve this problem, we used a chain of inverters as driver (Uyemura, 2001; Weste & Eshraghian, 1994).

Signal *a* (index of current ant) is connected to all of cells in ACO chip. For big problem, the capacitor of this node gets very large. We use H-Tree clock distribution structure for signal *a*, therefore in such a structure, the distances from the center to all branch points are the same and hence, the signal delays would be the same. However, this structure is difficult to implement in practice for big problem due to routing constraints and different fan-out requirements. In big TSP problem, the clock signals must be buffered in multiple stages to handle the high fan-out loads (Weste & Eshraghian, 1994).

We attempt to design a high speed ACO chip in here but for low speed chip (input clock lower than 100 KHz) many of mentioned problems are solved.

7. Conclusion

In this paper, an ACO chip is designed in 0.35 μm CMOS technology for the first time. A new ACO algorithm was presented which is suitable for CMOS implementation. This algorithm use sum of heuristics for select of best ant at the end of iteration. We used both heuristics value and pheromone value in ACO chip. Our chip converges in less iteration than last approaches because a new scheme for selection of next city is used. This chip is Mixed-Mode (Analog & Digital) chip and can be used for adaptive routing in telecommunication networks.

References

Aporntewan, C., & Chongstitvatana, P. (2001). Hardware implementation of the compact genetic algorithm. In *Proceedings of IEEE congress on evolutionary computation, South Korean*, pp. 624–629.

Dorigo, M., Birattari, M., & Stützle, T. (2006). Ant colony optimization artificial ants as a computational intelligence technique. *IEEE Computational Intelligence Magazine*(11), 28–38.

Dorigo, M., & Di Caro, G. (1999). The ant colony optimization meta-heuristic. In D. Corne et al. (Eds.), *New ideas in optimization* (pp. 11–32). London, UK: McGraw Hill.

Dorigo, M., Di Caro, G., & Gambardella, L. M. (1999). Ant algorithms for discrete optimization. *Artificial Life, 5*(2), 137–172.

Dorigo, M., Maniezzo, V., & Colorni, A. (1991). *Positive feedback as a search strategy*. Dipartimento di Elettronica, Politecnico di Milano, Italy, Tech. Rep. 91-016.

Dorigo, M., Maniezzo, V., & Colorni, A. (1996). Ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics — Part B, 26*(1), 29–41.

Gheysari, K., Khoei, A., & Mashoufi, B. (2009). Design a chip based on Ant Colony Optimization in CMOS technology. In *Computer society of Iran computer conference 2009 (CSICC 2009)*. Amirkabir University of Technology, Iran, March 2009.

Juang, C. F., & Chen, J. S. (2006). Water bath temperature control by a recurrent fuzzy controller and its FPGA implementation. *IEEE Transactions on Industrial Electronics, 53*(3), 941–949.

Juang, C. F., & Hsu, C. H. (2005). Temperature control by chip-implemented adaptive recurrent fuzzy controller designed by evolutionary algorithm. *IEEE Transactions on Circuits and Systems – I: Regular Papers, 25*(11), 2376–2384.

Juang, C. F., Lu, C. M., Lo, C., & Wang, C. Y. (2008). Ant colony optimization algorithm for fuzzy controller design and its FPGA implementation. *IEEE Transactions on Industrial Electronics, 55*(3), 1453–1462.

Jung, S., & Kim, S. S. (2007). Hardware implementation of a real-time neural network controller with a DSP and an FPGA for nonlinear systems. *IEEE Transactions on Industrial Electronics, 54*(1), 265–271.

Li, T. H. S., Chang, S. J., & Chen, Y. X. (2003). Implementation of human-like driving skills by autonomous fuzzy behavior control on an FPGA-based car-like mobile robot. *IEEE Transactions on Industrial Electronics, 50*(5), 867–880.

Merkle, D., & Middendorf, M. (2002). Fast ant colony optimization on runtime reconfigurable processor arrays. *Genetic Programming and Evolvable Machines, 3*, 345–361.

Nakao, M., & Izumi, K. (2004). Circuit designs and fabrication of swarm-intelligence LSIs based on modeling foraging behaviors of ants. In *The 47th IEEE International midwest symposium on circuits and systems*, pp. 97–100.

Razavi, B. (2000). *Design of analog CMOS integrated circuit*. McGrawHill Science Engineering.
Reinelt, G. (1991). TSPLIB – A traveling salesman problem library. *ORSA Journal on Computing, 3*, 376–384.
Scheuermann, B. (2005). *Ant colony optimization on runtime reconfigurable architectures*. Ph.D. thesis, Universität Fridericiana zu Karlsruhe.
Scheuermann, B., So, K., Guntsch, M., Middendorf, M., Diessel, O., ElGindy, H., et al. (2004). FPGA implementation of population-based ant colony optimization. *Applied Soft Computing*, 303–322.
Uyemura, J. P. (2001). *Introduction to VLSI circuits and systems*. Wiley, John & Sons.
Weste, N. E., & Eshraghian, K. (1994). *Principles of CMOS VLSI design* (2nd ed.). Addison-Wesley publishing Co.