Optimized FPGA Routing using Soft Computing

Saveena
P.T.U.Jalandhar
Computer Science & Engg.
D.A.V.I.E.T Jalandhar

Vinay Chopra
P.T.U.Jalandhar
Computer Science & Engg.
D.A.V.I.E.T Jalandhar

Dr. Amardeep Singh
Reader, Punjabi University Patiala
Computer Science & Engg.
UCOE Patiala

ABSTRACT
FPGAs are used for a wide range of applications, e.g. network communication, video communication and processing and cryptographic applications. It has been shown that FPGAs are suitable for the implementation of soft computing techniques like Neural Networks and Genetic Algorithms. In this work we have shown that Ant Colony Optimizations can also be implemented on FPGAs, leading to significant speedups in runtime compared to implementations in software on sequential machines. This paper presents an ant colony optimization algorithm for geometric FPGA routing for a route based routing constraint model in FPGA design architecture.

Keywords
Ant colony optimization, Boolean Satisfiability, Field Programmable Gate Arrays, Soft Computing

1. INTRODUCTION
By providing programmable selection of alternate logic and routing structures, field-programmable gate arrays can be considered to be located at the intersection of software and hardware-oriented systems. Using modern design software, circuits can be designed and implemented very rapidly, thereby avoiding the up-front cost of designing custom circuits and providing a quick means of correcting design errors.

After configuring the circuit onto the FPGA chip, it is switched into operational mode, upon which the inherent parallelism and pipelined design style can offer considerable speedup over instruction stream processors. FPGAs facilitate system development and allow easy and quick design changes and verification. Not only are they suitable for rapid prototyping, they can also substitute for standard logic and gate array solutions in small and medium volume productions. However, their high level of flexibility demands additional switches and routing, which in turn increase circuit delays and chip area relative to custom fabricated circuits. The layout structure of these FPGAs depends upon three parameters configurable logic blocks, I/O blocks and programmable routing.

Our main consideration in this paper is on programmable FPGA routing shown in [Section 2]. Boolean-based routing is a recent approach that is used for solving routing problem in FPGA layout. Boolean based routing problem can be represented as a large atomic Boolean function, which is satisfiable if the layout is routable otherwise routing option is not considered i.e. any satisfying assignment to the variables of the routing Boolean function represents a legal routing solution recent advances in SAT solving algorithms (learning and non-chronological backtracking) and efficient implementation techniques (e.g. fast implication engine) have dramatically improved the efficiency.

But there is still need to improve it so that FPGA routing task can be optimized. In this paper, we adapt an ACO algorithm to field programmable gate arrays (FPGAs). To the best of our knowledge, this is the first implementation of ACO for solving FPGA routing. The ant colony optimization meta-heuristic is adopted from the natural foraging behavior of real ants and has been used to find good solutions to a wide spectrum of combinatorial optimization problems. Classically many search style solutions have been proposed for SAT, the best known being variations of the Davis-Putnam procedure. It is based on a backtracking search algorithm illustrating that, at each node in the search tree, elects an assignment and prunes subsequent search by iteratively applying the unit clause. The other algorithms that are used in the SAT based problems are backtracking search, resolution based checker, integer linear programming based routing, BDD, recursive learning etc. A new approach for FPGA routing, which is improvement over other SAT solvability algorithms for FPGA routing is illustrated in this paper.

2. FPGA Layouts

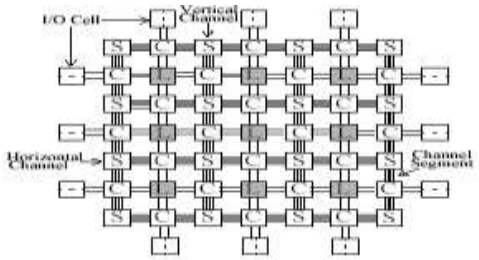
Standard island style FPGA architecture Xilinx 4000 [XILINX, 01] is used in this experiment and results are compared with the other SAT solvers, which are previously applied to this architecture. This is one of the most commonly used layout models in FPGA applications [see Fig. 1(a)]. It consists of two-dimensional array of configurable logic blocks CLBs, Connection blocks C blocks and switching blocks S blocks. Each CLB marked L in [Fig. 1(a)] contains the combinational and sequential logic that implements the functionality of a circuit. C and S blocks contain programmable switch form the routing resources. C blocks connect CLB pins to channels via programmable switches. S blocks are surrounded by C blocks and allow signals to either pass through.

A net consists of CLB pins that are interconnected with each other and can be decomposed into one or more horizontal and/or vertical net segments each of which is an alternating sequence of C and S blocks forming an uninterrupted path. A detailed route of a net is a set of wire segments and routing switches, within the

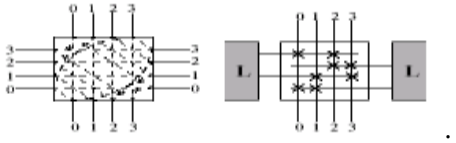
assigned routing area set by the global router. For each net segment, a detailed router assigns wire segments and routing switch following the topology specified by the global router such that no overlapping among detailed routes of different nets occurs.

The routing capacity of a given FPGA architecture is mainly expressed by three parameters W, Fc, Fs. The channel width is the number of tracks in a vertical or horizontal channel. The C block flexibility Fc is defined to be the number of tracks that each logic pin can connect to. The S block flexibility Fs denotes the number of other tracks that each wire segment entering an S block.

Each wire segment entering this S block can connect to one track on each of the other three sides, hence Fs=3[see Fig. 1(b)]. Each logic pin can be connected up to any two tracks in the C block, thus Fc=2[In Fig. 1(c)]



(a)



(b) $F_s = 3$ (c) $F_c = 2$

Figure1: Island style FPGA model

2.1 Boolean SAT based FPGA detailed routing formulation

In this approach geometric FPGA routing task is transformed into a Boolean satisfiability (SAT) equation with the property that any assignment of input variables that satisfies the equation specifies a valid route. The satisfiability equation is then modeled as Constraint Satisfaction problem, which helps in reducing procedural programming. Satisfying assignment for particular route will result in a valid routing and absence of a satisfying assignment implies that the layout is unroutable. In second step ant colony optimization algorithm is applied on the Boolean equation for solving routing alternatives utilizing approach of hard combinatorial optimization problems for stationary and non-stationary environments. The ACO based solution to SAT is then compared with the other SAT solver algorithms such as zChaff and GRASP. Preliminary experimental results suggest that the developed ant colony optimization algorithm is taking mlogm iterations, where m is number of Boolean instances and using extremely short CPU time finds all possible routes even for large FPGA circuits.

The present work is based on the optimization technique that could be further extended in several other optimization problems in the testing field such as vector re-ordering, scan chain partitioning, vector compaction, sequential test pattern generation etc.

Boolean SAT-based routing transforms the geometric routing task into a Boolean Satisfiability (SAT) problem by rendering the routing constraints as an atomic Boolean function. The generated Boolean function is satisfiable (has an assignment of input variables such that the generated function evaluates to constant "1") if and only if the design is routable. Any satisfying assignment to the binary variables of the Boolean function represents a legal routing solution. Moreover, by demonstrating the absence of satisfying assignments for a generated routing Boolean function, we can prove that no routing solution exists. A particular virtue of this method is that much of the geometric complexity of the interaction among objects (i.e., nets in routing) is hidden and rendered implicitly in the Boolean constraint functions so that all the objects are considered simultaneously. In other words, Boolean-based routing is a concurrent method allowing higher degrees of freedom for each object in contrast to the conventional one net-at-a-time approach.

In spite of these unique properties, the use of Boolean Satisfiability to solve VLSI routing is not as common as other methods using integer linear programming or heuristic search. To the best of our knowledge, was the first to establish a link between geometric layouts and Boolean formulations. In this work, he proves that a general dogleg channel routing problem belongs to the NP-complete class by reducing the 3-satisfiability problem to it, i.e., given a 3-satisfiability formula, he showed how to construct an instance of the channel routing problem that can be routed in a certain number of tracks if and only if the original formula is satisfiable. Capitalizing on this idea, [9] devised a formulation of conventional 2-layer channel routing as a generic Boolean SAT problem by encoding the information present in a channel's vertical constraint graph, horizontal constraint graph, and the anticipated channel width into Boolean constraint formulas on a set of n-bit Boolean vectors, one per net to be routed. Thus, if the generated Boolean formula is satisfiable, then any satisfying assignment corresponds to a feasible routing of the channel; otherwise (i.e., the function is proven to be unsatisfiable) the channel is provably unroutable with the anticipated channel width. Later, showed that the grid-based channel routing with the restricted two-layer model is a fixed-parameter tractable problem which can be solved in linear time with the fixed channel length. The idea was to record a net which leaves each track per column while sweeping columns from left to right in the channel.

The core idea of applying a Boolean SAT technique to a simple routing problem is illustrated in Fig. 2. It is a channel routing problem with four nets labelled A, B, C, and D (Fig. 2a). The goal is to assign a track number to each net such that distinct nets are nonoverlapping both horizontally and vertically. The channel routing problem is mapped into two matrices, one represents the horizontal overlapping and second represents vertical overlapping (Fig. 2b). Two types of constraints are defined to guarantee a legal channel routing solution. First, an exclusivity constraint insures that nets whose horizontal spans overlap are

assigned to different tracks. This constraint is typically represented by a horizontal constraint graph and can be conveniently expressed as a Boolean function (Fig. 2d). The other constraint insures that, for any two nets with pins in the same column on opposite sides of the channel, the net associated with the top pin is assigned a higher track number. This constraint is conveniently captured by the vertical constraint graph (VCG) shown in Fig. 2c, which in turn is equivalent to the Boolean function V of Fig. 2e. The conjunction (AND) of these two functions is the complete routability constraint Boolean function R for the channel (Fig. 2f).

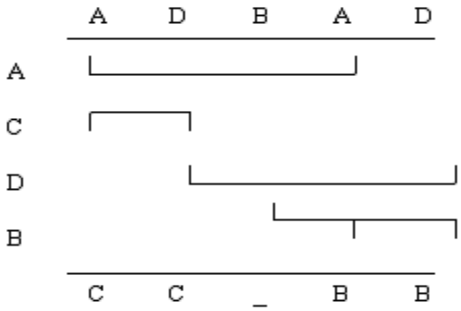


Fig. 2.a Channel to be routed

	A	B	C	D
A	0	1	1	1
B	0	0	0	0
C	0	0	0	1
D	0	1	0	0

	A	B	C	D
A	0	1	1	0
B	0	0	0	0
C	0	0	0	0
D	0	1	1	0

Fig. 2.b Matrix representation of Channel routing problem

$$E= (A\neq C) \Lambda (A\neq D) \Lambda (A\neq B) \Lambda (C\neq D) \Lambda (D\neq B)$$

Fig. 2.c Exclusivity Constraint

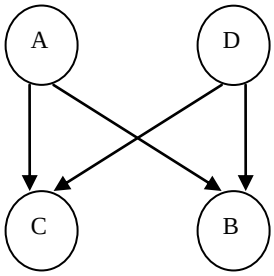


Fig. 2.d Vertical Constraint Graph (VCG)

$$V= (A>C) \Lambda (A>B) \Lambda (D>C) \Lambda (D>B)$$

Fig. 2.e Vertical Ordering Constraint

$$R=E \Lambda V$$

Fig. 2.f Channel routability constraint

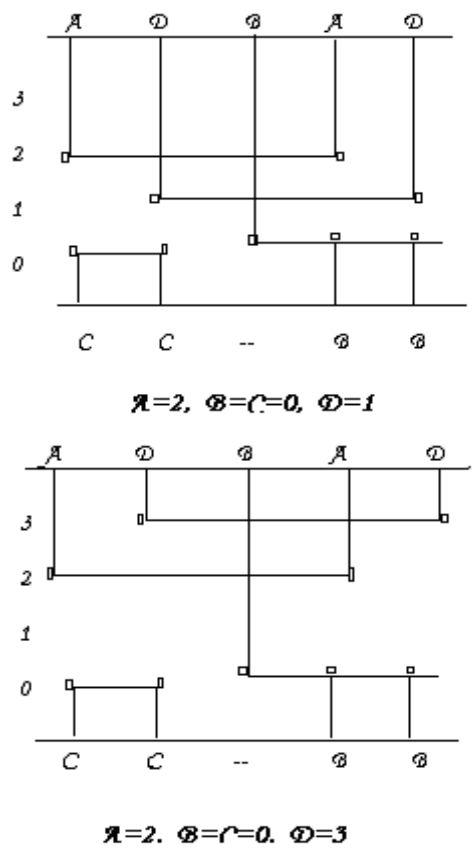


Fig. 2.g Two feasible solutions

Fig. 2. Boolean SAT modeling of channel routing problem.

Any track number assignment that makes R=1 corresponds to a feasible routing solution and completely specifies the net-to-track mapping. Two feasible assignments are shown in Fig. 2g.

Boolean-based routing, in general, has the following advantages over conventional one-net-at-a-time routing approaches:

- Simultaneous net embedding: The conventional one-net-at-a-time routing approach is notorious for being dependent on net ordering because previously routed nets act as obstacles to the yet-to-be-routed nets. In Boolean-based routing, all routing constraints are considered concurrently by a Boolean SAT solver, making net ordering irrelevant.
- Routability decision: The unsatisfiability of the routing Boolean constraint function, as proven by a Boolean SAT solver, directly implies that there is no feasible routing solution with the given placement and global routing configuration. On the other hand, any assignment to the Boolean variable vector that satisfies the routability Boolean function corresponds to a complete feasible detailed routing solution.

However, Boolean SAT-based routing is known to be less scalable than conventional routing approaches. In other words, the size of problems that can be attacked by Boolean SAT-based routing is smaller than that of conventional routing.

3 Ant Colony Optimization

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 [1,2]. Ant colonies are capable of finding shortest paths between their nest and food sources. This complex behavior of the colony is possible because the ants communicate indirectly by disposing traces of pheromone as they walk along a chosen path. Following ants most likely prefer those paths possessing the strongest pheromone information, thereby refreshing or further increasing the respective amounts of pheromone. Since ants on short paths are quicker, pheromone traces on these paths are increased very frequently. On the other hand, pheromone information is permanently reduced by evaporation, which diminishes the influence of formerly chosen unfavorable paths. This combination focuses the search process on short, favorable paths.

Inspired by this biological paradigm, Dorigo et al. [3–5] introduced a metaheuristic known as ant colony optimization (ACO). In ACO, a set of artificial ants searches for good solutions for the optimization problem under consideration. Each ant constructs a solution by making a sequence of local decisions. Its decisions are guided by pheromone information and some additional heuristic information (if applicable).

After a number of ants have constructed solutions, the best ants are allowed to update the pheromone information along their path through the decision graph. Evaporation is accomplished by globally reducing the pheromone information by a certain percentage. This process is repeated iteratively until a stopping criterion is met. ACO has shown good performance on several combinatorial optimization problems, including scheduling [6], vehicle routing [7], constraint satisfaction [8], and the quadratic assignment problem [9].

3.1 Description

The objective of ACO is to find good solutions for a given combinatorial optimization problem. The problems considered usually allow solutions to be expressed as a permutation π of n given items. The pheromone information is encoded in an n x n pheromone matrix [τij]. Depending on the problem the pheromone value τij expresses the desirability to assign item j to place i of the permutation (place x item coding) or the desirability problem, n given jobs have to be scheduled onto a single machine. For every job j, its deadline dj and the processing time pj are given. If Cj denotes the completion time of job j in a schedule, then Lj = Cj − dj defines its lateness and Tj = max(0,Lj) its tardiness.The objective is to find a schedule minimizing the total tardiness of all jobs ∑ j=0 to n−1Tj. Since the relative position of a job in the schedule is more important than its predecessor or successor in the schedule a place x item pheromone matrix is used.

• For the quadratic assignment problem (QAP), n facilities, n locations, and two n x n matrices [dij] and [fhl] are given, where dij is the distance between locations i and j and fhl is the flow between facilities h and l. The goal is to find an assignment of facilities to locations, i.e. a permutation π of [0, n−1], such that the sum of distance-weighted flows between facilities ∑ j,i=0 to n−1dπ(i)π(j)fij is minimized. Solutions are constructed by successively assigning facilities to places (locations) in the permutation, which means that, like for SMTTP, a place x item pheromone matrix is used.

• For the traveling salesperson problem (TSP), n cities are given with distances dij between cities i and j for i; j ϵ [0, n−1]. The goal is to find a distance minimal Hamiltonian cycle, i.e. a mono-cyclic permutation π of [0: n−1] which minimizes ∑i=0 to n-1 diπ(i). Since the neighborhood of cities in the permutation is important for this problem an item x item pheromone matrix is used.

Typically, when constructing a solution, ants do not rely solely on pheromone information, but also have access to some heuristic information ηij , which signifies the immediate impact that a local decision might have on solution quality. For example, in TSP the heuristic value of choosing to visit city j after the last chosen city i is considered to be inversely proportional to the distance separating them, ηij = 1 = dij.

The standard ACO algorithm (see Fig. 2) starts by initializing the pheromone matrix, setting every pheromone entry to an initial value τinit > 0. For problems with an item x item encoded pheromone matrix, e.g. TSP, the pheromone entries on the diagonal are set to 0, since no city can be its own successor/ predecessor. In every iteration of the algorithm, m ants generate solutions π0, . . . , πm−1. An ant builds a solution by making a sequence of local decisions, i.e. successive selections of items. Every decision is made randomly according to a probability distribution over the so far unchosen items in selection set S:

$$\forall j \in S : p_{ij} = \frac{\tau_{ij}^{\alpha} \eta_{ij}^{\beta}}{\sum_{l \in S} \tau_{il}^{\alpha} \eta_{il}^{\beta}}$$

where parameters α and β are used to determine the relative influence of pheromone values and heuristic to position item *j* immediately after item *i* in the permutation (item x item coding). Three examples for combinatorial optimization problems and the corresponding types of pheromone encoding are given below:

• For the single machine total tardiness values. Initially, the selection set S contains all items; after each decision, the selected item is removed from S. Every solution is evaluated according to the respective objective function. After m solutions have been generated, the solution qualities are compared to determine the best solution π∗ of the current iteration.

The pheromone matrix is then updated in two steps:

(1) Evaporation: All pheromone values in the matrix are reduced by a relative amount: for all i, j ϵ [0, n− 1] : τij → (1 − ρ)τij.

(2) Intensification: The pheromone values along the best solution π∗ are increased by an absolute amount: ∀i ∈ [0, n − 1] : τiπ∗ (i) → τiπ∗ (i) + Δ.

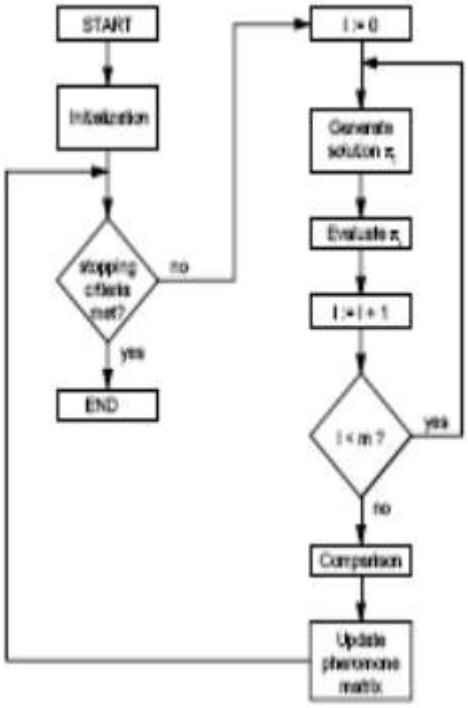


Fig. 3: ACO Processing Flow

4. The Proposed method: An ACO Based Detailed FPGA Routing

In this paper we have proposed two tier approaches for the FPGA routing solution. First, geometric FPGA detailed routing task is solved by transforming it into a Boolean satisfiability equation with the property that any assignment of input variables that satisfies the equation specifies a valid routing. Satisfying assignment for particular route will result in a valid routing and absence of a satisfying assignment implies that the layout is un-routable. In second step, Ant colony earch algorithm is applied on this Boolean equation for solving routing alternatives utilizing the properties of ACO. The simulated results are satisfactory and give the indication of applicability of ant colony optimization for solving the FPGA Routing problem.

To solve a constraint satisfaction problem, we can explore the search space entirely until a solution is found or it is proven that no solutions exist. These are called complete approaches and these are usually combined with some filtering or propagation techniques to reduce the running time of the algorithm. Completeness is desirable in that the success rate improves. This is because the entire search space is explored and all solutions, if any can be found. However, the inherently intractable nature of CSP's makes completeness approaches inefficient, especially for large instances of hard combinatorial problems. This is one of the main reasons for the emergence of stochastic local search and evolutionary approaches to solve CSP's. The algorithm that we use is based on the Ant Colony Optimization paradigm.

The standard ACO algorithm starts by initializing the pheromone matrix, setting every pheromone entry to an initial value $\tau init > 0$. For this problem with MxM encoded pheromone matrix, the pheromone entries on the diagonal are set to 0, since no net can be its own successor/ predecessor. In every iteration of the algorithm, m ants generate solutions $\pi 0, \ldots, \pi m-1$. An ant builds a solution by making a sequence of local decisions, i.e. successive selections of items. Every decision is made randomly according to a probability distribution over the so far unchosen items in selection set. Initially, the selection set contains all items; after each decision, the selected item is removed from selection set. Every solution is evaluated according to the respective objective function. After m solutions have been generated, the solution qualities are compared to determine the best solution $\pi*$ of the current iteration.

The pheromone matrix is then updated in two steps:

- *Evaporation*: All pheromone values in the matrix are reduced by a relative amount.
- *Intensification*: The pheromone values along the best solution $\pi*$ are increased by an absolute amount.

The ACO algorithm executes a number of iterations until a specified stopping criterion has been met, e.g. a predefined maximum number of iterations has been executed, a specific level of solution quality has been reached, or the best solution has not changed over a certain number of iterations.

5. CONCLUSION

We have tried to improve the performance of the FPGA routing by solving it using ACO algorithm Our results have shown that the algorithm is taking $O(nm/\rho \log n)$ running time, which is an optimal solution. Our algorithm works as a collection of agents work collaboratively to explore the different routes. A stochastic decision making strategy is proposed in order to combine global and local heuristics to effectively conduct this exploration. Our algorithm is more effective in finding the near optimal solutions and scales well as the problem size grows. It is also shown that with substantial less execution time the proposed method achieves better solutions than the popularly used zChaff and GRASP approach.

REFERENCES

[1] S. Bade, B. Hutchings, "Fpga based stochastic neural network implementation, in: Proceedings of the IEEE Workshop on FPGAs for Custom Computing Machines", 1994, pp. 189–198.

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

[3] E.-G. Talbi, O. Roux, C. Fonlupt, D. Robillard, "Parallel ant colonies for combinatorial optimization problems", in: J.R. et al. (Eds.), Parallel and Distributed Processing, 11 IPPS/SPDP'99 Workshops, no. 1586 in LNCS, Springer-Verlag, 1999, pp. 239–247.

[4] Stützle, T. and H. H. Hoos, MAX-MIN ant system, Future Gener. Comput. Syst., vol. 16, no.8, pp.889-914, 2000.

[5] M. W. Moskewicz, C. F. Madigan, Y. Zhao, L. Zhang, and S. Malik. Chaff: Engineering an efficient SAT solver. In Proceedings of the 38th Design Automation Conference (DAC'01), pages 530–535, June 2001.

[6] E. Goldberg and Y. Novikov, "BerkMin: A Fast and Robust SAT solver," Proc. Design, Automation, and Test in Europe (DATE '02, pp. 142-149, Mar. 2002.

[7] Stützle, T. and Dorigo M., "A Short Convergence Proof for a Class of ACO Algorithms", IEEE Transactions on Evolutionary Computation, 6 (4), 2002 (in press).

[8] Niklas E´en and Niklas S¨orensson. An extensible sat solver. In Proceedings of the Sixth International Conference on Theory and Applications of Satisfiability Testing, LNCS 2919, pages 502–518, 2003.

[9] Neumann, F. and Witt, C., Runtime Analysis of a Simple Ant Colony Optimization Algorithm. Electronic Colloquium on Computational Complexity (ECCC), Report No. 84.,2006.

[10] Eliezer L. Lozinskii, Impurity: Another phase transition of SAT, Journal on Satisfiability, Boolean Modeling and Computation, vol. 1, 2006, pp. 123-14.

[11] Ines Alaya, Christine Solnon, Khaled Ghedira. "Ant Colony Optimization for Multi-objective Optimization Problems", ICTAI 2007 vol. 1, pp. 450-457, 2007

[12] Walter J. Gutjahr. "First Steps to the Runtime Complexity Analysis of Ant Colony Optimization", Computers and Operations Research, Volume 35, Issue 9, pp. 2711-2727, 2008

[13]Nattapat Attiratanasunthron Jittat Fakcharoenphol, "A Running Time Analysis for an Ant Colony Optimization Algorithm for Shortest Paths on Directed Acyclic Graphs", Information processing letters, vol 105, Issue 3, pp. 88-92, 2008.