CONSTRUCTION SCHEDULING WITH ARTIFICIAL AGENTS AND THE ANT COLONY OPTIMIZATION METAHEURISTIC

Symeon Christodoulou[1]

[1]*Department of Civil and Environmental Engineering, University of Cyprus, P.O. Box 20537, 1678 Nicosia, Cyprus*

The research outlined in this paper aims the development of a methodology to arrive at critical path calculations in construction networks using Ant Colony Optimization (ACO) algorithms. Ant Colony Optimization is a population-based, artificial multi-agent, general-search technique for the solution of difficult combinatorial problems. The method's theoretical roots are based on the behaviour of real ant colonies and the collective trail-laying and trail-following of its members in searching for optimal solutions in traversing multiple paths. In essence, ACO is inspired by the foraging behaviour of natural ant colonies which optimize their path from an origin (ant nest) to a destination (food source) by taking advantage of knowledge acquired by other ants that previously traversed the possible paths. In computer implementations of the ACO algorithms, artificial ants are both agents and solution-construction procedures that stochastically build solutions by considering (1) artificial pheromone trails which change dynamically at run time to reflect the agents' acquired search experience, and (2) heuristic information on the problem/network being solved. The paper outlines the fundamental mathematical background of the ACO method and a suggested possible implementation strategy for solving for longest (critical) paths in construction schedule networks.

Keywords: ant colony optimization, construction scheduling, critical path.

INTRODUCTION

Currently, the most widely used method for construction scheduling is the critical path method (CPM) which, in essence, is based on forward and backward passes of the activity network to arrive at longest path and total duration calculations. The method provides planners and constructors with a formalized way of calculating the earliest and latest dates each construction activity can start and finish at, subject to the envisioned network topology (activity relationships) and the imposed time, resource and logic constraints.

Undoubtedly, over the years CPM has grown in functionality, accessibility and acceptability. Yet, some limitations of the method have recently given rise to the search for complimentary methodologies that could address some of either the perceived limitations or the desired additional features.

Among the most notable limitations are: (i) the inability of CPM-based tools to calculate longest (or shortest) paths from a node to any node, (ii) the inability of CPM-based tools to account for resource-driven activity relationships ("AND"/"OR" combinations of resources), and (iii) the computational inefficiency of the critical path method.

[1] schristo@ucy.ac.cy

Item (i) refers to the need to identify the longest (or shortest) path to a desired activity from any other activity in a project network. This feature would enable constructors identify the paths to start (or completion) of any desired activity from any other activity in the network and act accordingly to recover or accelerate the project and to redirect resources in the most efficient manner. Similarly, this added feature would enable planners and managers interested at successful timely completion of specific activities identify the longest, or shortest, path to that activity and mediate associated risks as needed.

Item (ii) refers to the need to account for resource-based activity relationships. In this scenario, tasks are performed and schedule advancement is made upon additive or conditional combinations of resources. Even though CPM schedules can nowadays be “resource-loaded”, so that resources are accounted for during the network calculation phase, such schedules do not enable accounting for real-life situations where activity sequencing is based on resource (rather than activity) relationships. In other words, how can a planner account for the situation where an activity can start only when a resource from a predecessor activity is freed and then combined with a resource from another predecessor activity to enable the start of the successor activity without completion of the two predecessor activities?

Item (iii) refers to the perceived computational inefficiency of the traditional CPM algorithms. If one considers that solution of the activity network can be achieved by solving the underlying equations which represent the activity interrelationships, then possible solution methodologies could include solution of linear equations, linear programming, and forward/backward passes. The first two approaches imply that the applied solver is capable of handling a large number of equations and constraints, as well as optimization. Despite that scheduling problems within certain categories in general have polynomial-type exact solution algorithms and their solution is simple, in its most general form, though, resource-constrained scheduling problems are “NP-hard”, meaning that there are no known algorithms for finding optimal solutions in polynomial time. This class of problems (NP-class) requires computational power that increases exponentially with the size of the problem and has no exact solution. In such cases a heuristic approach is warranted. The last approach implies exhaustive enumeration of the possible paths in a project network, and calculation through network-traversing. In essence, calculation in this case is achieved by starting from the first activity and exhaustively identifying all possible paths (successive activities) until reaching the last activity, adding the duration of each link to the total duration of the identified path up to that point (*EarlyStart*, *EarlyFinish* dates) and then identifying the longest path (forward pass). A reverse pass (same exhaustive approach) is used to calculate the latest dates each activity can start and finish subject to keeping the project end date fixed (as calculated during the forward pass phase). The combination of the two network passes provides the *TotalFloat* of each activity (*LateStart – EarlyStart*, or *LateFinish – EarlyFinish*) and therefore the critical path (critical are the activities with *TotalFloat* = 0). In summary, despite the methods’ relative ease of application they possess inefficiencies (i.e. the exhaustive enumeration of network paths) and computational constraints (i.e. the maximum number of equations and inequalities the applied solver can solve).

ANT COLONY OPTIMIZATION

Introduction

Ant Colony Optimization (ACO) is a population-based general search technique inspired by the foraging behaviour exhibited by real ant colonies. The method was first proposed by Dorigo (1991, 1992) for the solution of difficult combinatorial problems, and further expanded upon by several other researchers (Dorigo et al. 1996, Maniezzo et al. 2004). The method is characterized by "*the combination of a priori information about the structure of a promising solution with posterior information about the structure of a previously obtained good solution*" (Maniezzo et al. 2004), an attribute that is very suitable to network traversing and thus construction scheduling. In essence, the method uses knowledge acquired by one artificial agent during its path-searching, when constructing the next feasible or optimal solution of a given path-traversing problem.

The underlying methodology is modelled after the behaviour exhibited by real-life ant colonies as they search for optimal solutions in node-to-node path traversing situations, during which a shortest path in a static or dynamic topology is sought. The behaviour exhibited by such ants is characterized by a reinforcement mechanism that helps steer succeeding ants to the most frequently previously traversed path. In particular, an ant randomly traversing possible paths can help find the shortest path between food sources and a nest and in doing so deposit a chemical substance called "*pheromone*", forming "*pheromone trails*" which can then be followed by other ants in the colony. When choosing their way through the possible path routes, ants smell the deposited pheromone and tend to follow those paths marked by stronger pheromone concentrations. Therefore, while an isolated ant moves essentially at random, an ant encountering a previously traversed path and pheromone-laid trail can detect such, decide with high probability to follow it and subsequently reinforce the trail with its own pheromone.

The collective behaviour is therefore characterized by a positive (reinforcing) feedback loop where the probability with which each ant chooses the path to follow increases with the number of ants having chosen the same path in the preceding steps. The pheromone trail is reinforced with each successive pass until the ant population and path traversing converge to the shortest path between source and destination, and the final result is the relatively quick convergence of the path-traversing to the shortest path.

Theoretical Framework of the ACO Metaheuristic

A number of ACO algorithms, starting from the original work by Dorigo (1991, 1992), have been developed and proposed over the years. The common framework for ACO applications was proposed posteriori to be the ACO metaheuristic (Dorigo, Maniezzo and Colorni 1999), with artificial ants seen as stochastic solution procedures and acting as agents. The solution construction is biased by the pheromone trails which change at run-time, the heuristic information on the problem instance and the ants' private memory.

The generic problem topology was outlined by Stützle and Dorigo (2002). It consists (a) of a finite set of components, C, (b) a set of problem states, x, defined in terms of sequences (relationships) over the elements of C, (c) a set of all possible sequences, denoted by X, and (d) a finite set of constraints in the system, Ω, which defines the feasible states and the set of feasible solutions, S^*, which are a subset of the feasible states. Furthermore, a cost function $f(s,t)$ can be associated with each candidate solution, *s*, and in some cases a separate cost function is defined and associated to states other than solutions.

The generic behavior of the artificial ants was also outlined by Stützle and Dorigo (2002) as follows:

- Ants build solutions by moving on the construction graph *G=(C, R)*, where *C* is the set of components in the network, and *R* is the set of relationships (connections) fully connecting the components. Even though both feasible and infeasible solutions can be built, artificial ants, in general, try to build feasible solutions. The problem constraints, Ω, are implemented during the network-traversing and the policy followed by the artificial ants.
- The components, $c_i \subseteq C$, and connections, $r_{ij} \subseteq R$, can have a pheromone trail, τ, associated with them which allows for the implementation of a long-term memory policy about the ant search process. Similarly, the components, $c_i \subseteq C$, and connections, $r_{ij} \subseteq R$, can have a heuristic value, η, which allows incorporation of problem-specific information.
- The path that each artificial ant, *k*, follows can be stored in the ant's memory M^k.
- Each artificial ant, k, can be assigned a start state, x_s^k, and one or more termination conditions, e^k. The construction procedure of ant k stops when at least one of the termination conditions e^k is satisfied.
- When in state $x_r = (x_{r-1}, i)$ an ant attempts to move to any node j in the feasible solutions subset (immediate successors) N_i^k. If this is not possible, it might be allowed to move to any other node that it is not part of the immediate-successors subset.
- The move to a successor node is determined by a stochastic decision rule and it is subject to a function of the locally pheromone and the connection's heuristic, the ant's memory, and the problem constraints.
- Each addition of a component c_j to the current solution updates the pheromone trail associated with it.

Once a solution is built, the ant retraces the same path backwards and updates the pheromone trails of the used components or connections.

CONSTRUCTION SCHEDULING USING ANT COLONY OPTIMIZATION

Introduction

Construction scheduling exhibits many similarities to the ACO metaheuristic, since the underlying network topologies and path-searching approach to longest (or shortest) path calculations are comparable. If one substitutes the search for shortest path (ACO) to the search for longest path (CPM) and treats ACO ants, states, connections and cost function to CPM's resources, activities, relationships and durations respectively then the ACO metaheuristic can be employed in solving for the longest path in connected, acyclic graphs (such as construction activity networks).

Furthermore, by considering different nodal states and ant types then the search for longest path can be accompanied with calculations of longest path from a node to any node, as well as with resource optimization.

ACO-Based Algorithm

For a given construction network topology (project schedule) defined by a graph *G=(N,A)* with N being the set of nodes (activities) and A being the set of arcs (activity relationships) connecting the subject nodes, the proposed ACO-based procedure for finding the critical path(s) between chosen nodes N_1 and N_2 can be summarized by the following steps:

1. Initialize all arcs with small amount of pheromone, τ_0. This value can be an inverse line-distance between the nodes N_1 and N_2, or the inverse line-distance of the subject arc.
2. An artificial ant is launched from node N_1 (the start node) pseudo-randomly walking from a node to a successor node via the connecting arcs until it reaches either the end-node (N_2) or a dead end. When at a given node, the artificial ant's selection of an arc to follow is probabilistic, based on a stochastic assignment of each i^{th} arc's likelihood of selection, as defined by

$$p_i = \frac{\tau_i \eta_i^{\beta}}{\sum_i \tau_i \eta_i^{\beta}} \tag{1}$$

   In the above equation, τ_i is the pheromone concentration on the i^{th} arc, η_i is an a priori available heuristic value for the i^{th} arc and β_i is a parameter determining the relative influence of the heuristic information. The value of η_i can be defined either as the inverse of the length of the arc, or the inverse of the length of the arc plus the line-distance between the subject node and N_2. It should be noted that previously visited arcs are excluded from the selection (to enable complete "tree spanning" and avoid "memorization").
3. The selection is further assisted by the consideration of a randomly generated number, $0 \le q \le 1$, which is compared to a predefined value, q_0, specific to the network topology. If $q \le q_0$ then the arc with the highest value p_i is selected. Otherwise, a random selection of an arc is used based on the distribution defined by the equation for p_i.
4. Upon crossing each i^{th} arc during the aforementioned solution-constructing phase a local pheromone update rule is applied to update the level of pheromone concentration at the given arc. The updated pheromone level is defined by

$$\tau_i = (1 - \rho)\tau_i + \rho\tau_0 \tag{2}$$

   where ρ is another network topology parameter ($0 \le \rho \le 1$). As already noted, the goal of the local updating is to enable exploration of more path/route variations by making already traversed arcs less likely to be chosen again during the randomization of the arc selection process.

5. Steps (2) - (4) are repeated for all ants in the ant colony and the most successful ant (i.e. the one whose path defines the solution) is used to globally update the network's pheromone trails. The global update rule is defined by

$$\tau_i = (1-\alpha)\tau_i + \alpha\tau_L \tag{3}$$

where α is yet another network topology parameter ($0 \le \alpha \le 1$) whose value determines the level of evaporation of pheromone concentrations. The factor τ_L is a value inversely proportional to the path length of the best solution in case of an arc visited by the best ant or zero for all other ants.

6. The global update rule can be applied by either the "global-best" or the "iteration-best" ant. In the first case, the ant to perform the update is the one that obtained the best solution (found the longest path in the network) during the entire optimization process. In the second case the update is performed by the ant reaching the best solution during each iteration of the algorithm.

7. Steps (2) - (6) are repeated for either a fixed number of iterations or until a predefined condition is met, and upon termination of the algorithm the pheromone trail in the graph *G=(N,A)* is used to determine the solution (the arcs with highest pheromone concentration form the longest path of the network).

Case Study

The ACO algorithm was implemented by means of custom software (Christodoulou 2005). The software was designed and developed so as to be able not only to emulate traditional (legacy) construction scheduling applications and integrate with their underlying databases but to also be able to complement them with the ACO metaheuristic features. Furthermore, the developed software application can be executed in either "test mode" (randomized network topologies) or "project mode" (actual construction schedules to be solved) and be integrated with external database management systems to account for additional common construction features such as activity costs, resource types, etc. The ability to integrate with legacy scheduling software (such as Primavera's *Primavera Project Planner*, or Microsoft's *Project*), furnishes users with the ability to get the most out of CPM and ACO applications.

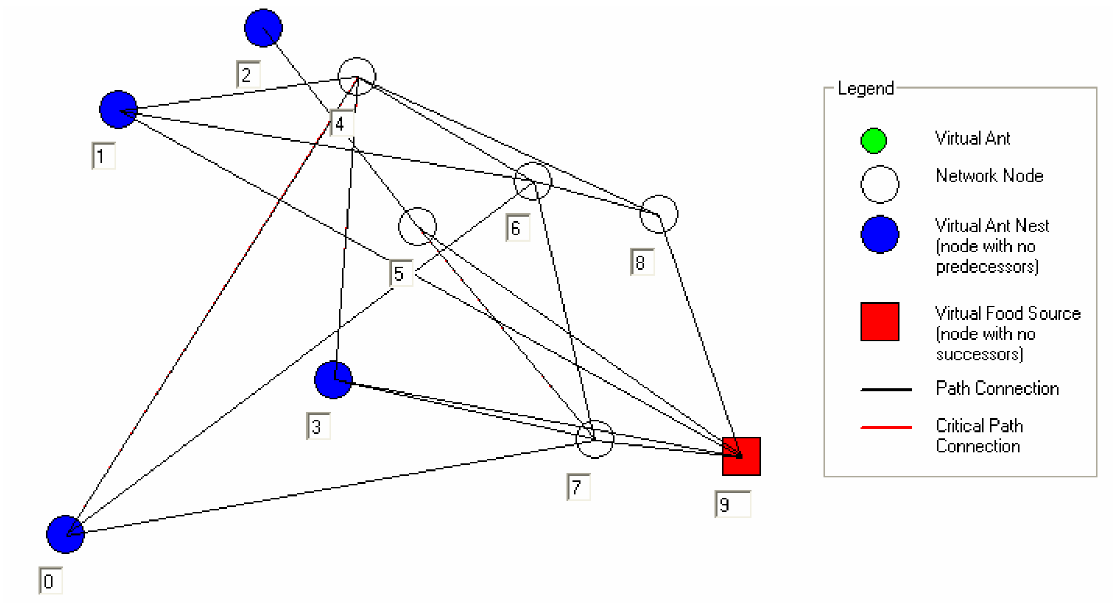

0
1
2
3
4
5
6
7
8
9
Legend
Virtual Ant
Network Node
Virtual Ant Nest (node with no predecessors)
Virtual Food Source (node with no successors)
Path Connection
Critical Path Connection

Figure 1: Topology of case study project network

The developed software was tested on several case studies, one of which is presented herein. The case study presented (Figure 1) is based on a randomized topology of 10 nodes, 18 node connections (a duplicate connection generated by the program was subsequently ignored), and assumed topology parameters of $q_0 = 0.5$, $\beta = 1.0$, $\rho = 0.5$, $\alpha = 0.5$, and $C = 50$. Of the 10 network nodes included in the assumed network topology, 4 are "ant nests" (i.e. nodes with no predecessor nodes), 5 are regular nodes and 1 is a "food source" (i.e. a node with no successor nodes).

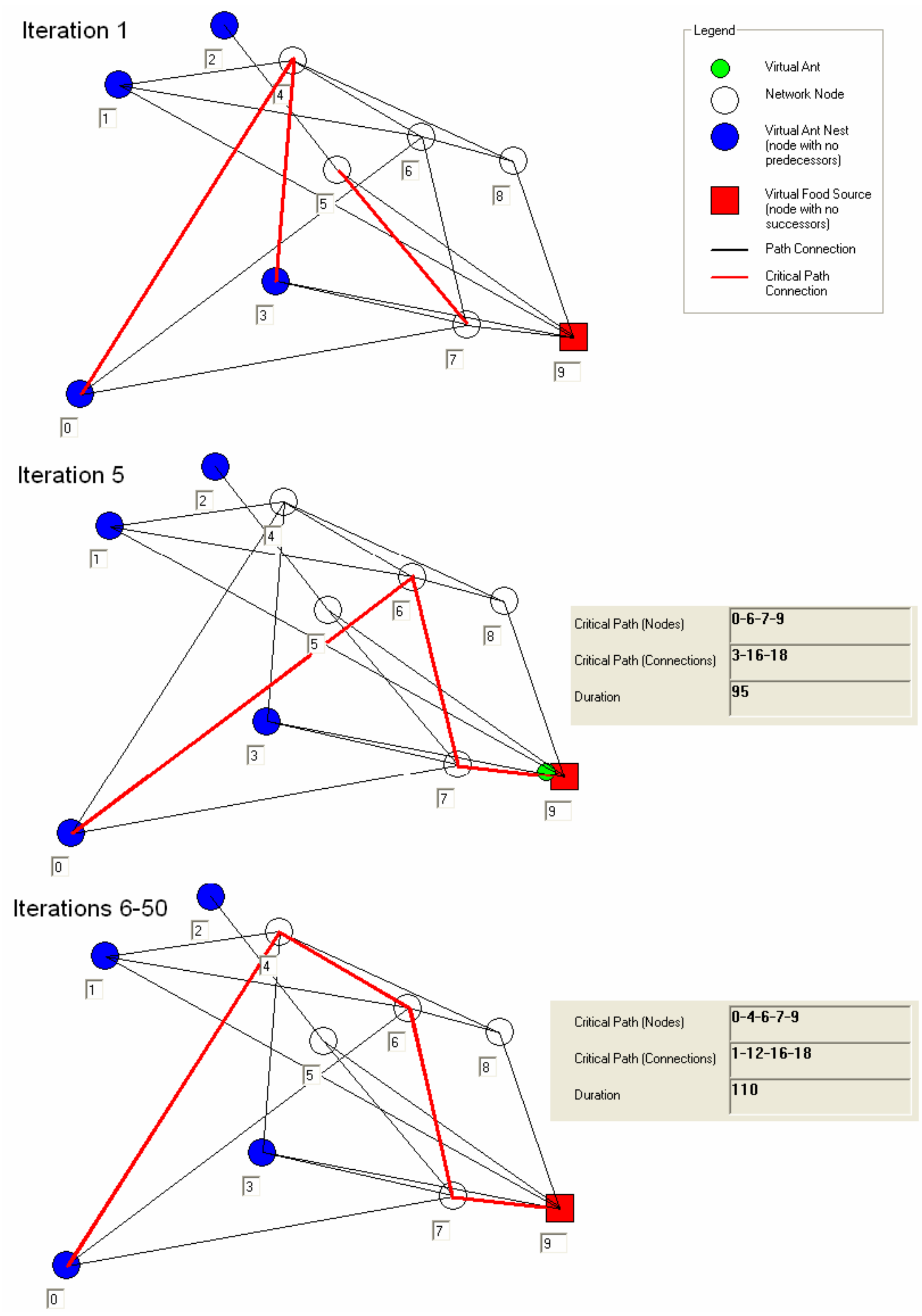
It is also noted that the generated network was based on unidirectional nodal connections (an acyclic graph) so as to emulate real-life construction networks, and that the network construction was based on an assumed maximum number of three successor connections per node, to simplify the manual calculations and subsequent verification by standard CPM procedures.

The critical path calculations on the case study topology, and the resulting *EarlyStart*, *EarlyFinish*, *LateStart*, *LateFinish* and *TotalFloat* values obtained by applying traditional CPM procedures are tabulated in Table 1. As shown, the CPM calculations result in identifying activities "0-4", "4-6", "6-7" and "7-9" as critical (*TotalFloat* = 0), thus indicating a total project duration of 52+19+25+14 = 110 time-units.

Table 1: Solution of the case-study network topology using the Critical Path Method (CPM)

Start Node	End Node	Duration	Successor Nodes	Early Start	Early Finish	Late Start	Late Finish	Total Float	Critical ?
0	4	52	6, 8	0	52	0	52	0	Yes
0	7	51	9	0	51	45	96	45	-
0	6	56	7, 8	0	56	15	71	15	-
1	4	23	6, 8	0	23	29	52	29	-
1	9	68	-	0	68	42	110	42	-
1	6	40	7, 8	0	40	31	71	31	-
2	5	24	7, 9	0	24	46	70	46	-
3	7	25	9	0	25	71	96	71	-
3	4	29	6, 8	0	29	23	52	23	-
3	9	40	-	0	40	70	110	70	-
4	6	19	7, 8	52	71	52	71	0	Yes
4	8	31	9	52	83	55	86	3	-
5	9	38	-	24	62	72	110	48	-
5	7	26	9	24	50	70	96	26	-
6	7	25	9	71	96	71	96	0	Yes
6	8	12	9	71	83	74	86	3	-
7	9	14	-	96	110	96	110	0	Yes
8	9	24	-	83	107	86	110	3	-

Application of the ACO methodology generates different topology states at the end of each iteration (as shown in Figure 2). Since the method is "intelligently" iterative (each successive iteration is selectively dependant on previously acquired knowledge about the topology) the critical path may change several times before converging and stabilizing to the correct solution (Figure 2). At each iteration, though, the algorithm generates the pheromone concentration levels (acquired knowledge on the criticality of each connection and node) and the resulting longest path (thus the critical activities).

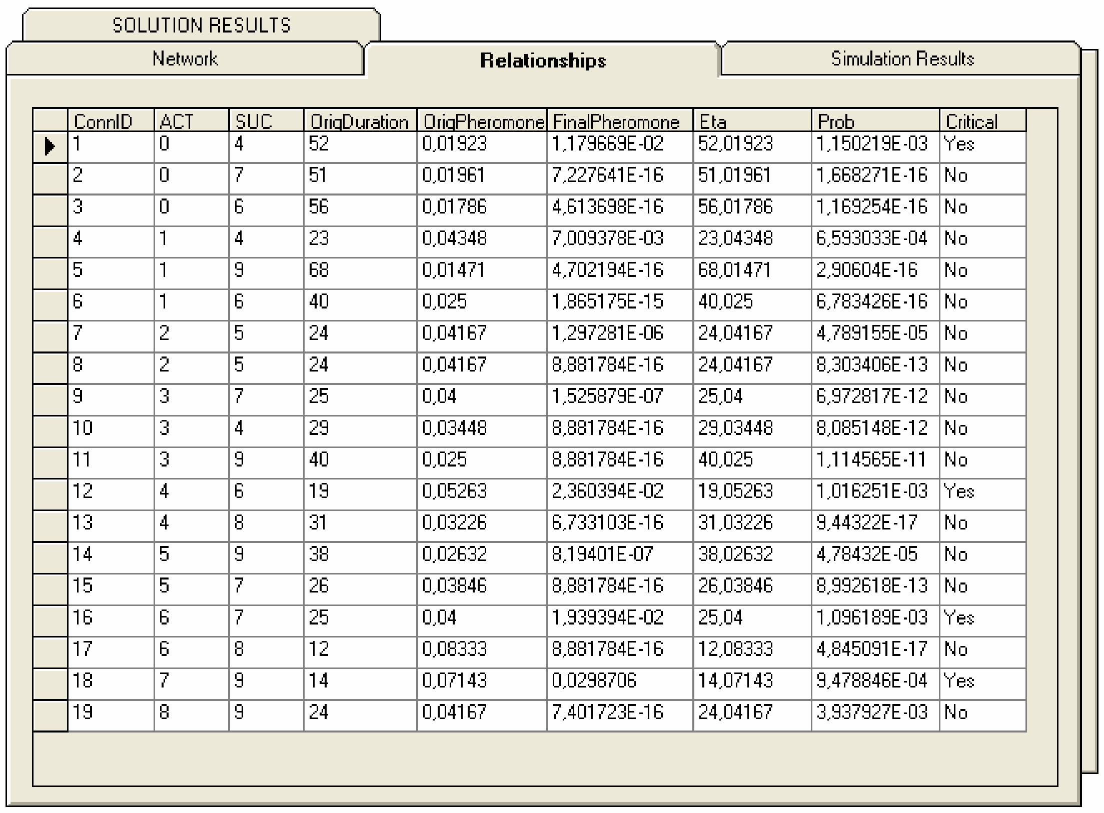


Critical Path (Nodes)	0-6-7-9
Critical Path (Connections)	3-16-18
Duration	95

Critical Path (Nodes)	0-4-6-7-9
Critical Path (Connections)	1-12-16-18
Duration	110

Figure 2: Network states at different ACO iterations

The final (solution) results obtained by the ACO algorithm are shown in Figure 3, tabulated as *FinalPheromone* values. The algorithm considers these values during the solution phase and decides which activities on a continuous path are critical, also outputting the "*Critical Path (Nodes)*", "*Critical Path (Connections)*" and "*Duration*" values (Figure 2). As seen, for the particular case study, the ACO algorithm identifies activities “0-4”, “4-6”, “6-7” and “7-9” (connections 1, 12, 16 and 18 respectively) as critical and calculates the critical path to be of 110 time-units total duration, which agrees with the results given by traditional CPM-based software.



ConnID	ACT	SUC	OrigDuration	OrigPheromone	FinalPheromone	Eta	Prob	Critical
1	0	4	52	0,01923	1,179669E-02	52,01923	1,150219E-03	Yes
2	0	7	51	0,01961	7,227641E-16	51,01961	1,668271E-16	No
3	0	6	56	0,01786	4,613698E-16	56,01786	1,169254E-16	No
4	1	4	23	0,04348	7,009378E-03	23,04348	6,593033E-04	No
5	1	9	68	0,01471	4,702194E-16	68,01471	2,90604E-16	No
6	1	6	40	0,025	1,865175E-15	40,025	6,783426E-16	No
7	2	5	24	0,04167	1,297281E-06	24,04167	4,789155E-05	No
8	2	5	24	0,04167	8,881784E-16	24,04167	8,303406E-13	No
9	3	7	25	0,04	1,525879E-07	25,04	6,972817E-12	No
10	3	4	29	0,03448	8,881784E-16	29,03448	8,085148E-12	No
11	3	9	40	0,025	8,881784E-16	40,025	1,114565E-11	No
12	4	6	19	0,05263	2,360394E-02	19,05263	1,016251E-03	Yes
13	4	8	31	0,03226	6,733103E-16	31,03226	9,44322E-17	No
14	5	9	38	0,02632	8,19401E-07	38,02632	4,78432E-05	No
15	5	7	26	0,03846	8,881784E-16	26,03846	8,992618E-13	No
16	6	7	25	0,04	1,939394E-02	25,04	1,096189E-03	Yes
17	6	8	12	0,08333	8,881784E-16	12,08333	4,845091E-17	No
18	7	9	14	0,07143	0,0298706	14,07143	9,478846E-04	Yes
19	8	9	24	0,04167	7,401723E-16	24,04167	3,937927E-03	No

Figure 3: Solution of sample network using the ACO algorithm

Variations and Extensions of Applied Method

Even though this paper presents longest-path calculations in construction networks and identification of longest path activities and project total duration, the applied ACO methodology can be modified and extended to account for resource and cash flows, and node-to-node longest path calculations. The former can be achieved by considering the respective activity assignments and including them in the cost function of each arc, used in the optimization process. The latter can be achieved by setting the start node of the node-to-node sequence of interest as “ant nest” and the end node as “food source” and reconstructing the solution path (longest path) while all other nodes are set as plain network nodes.

CONCLUSIONS AND FUTURE WORK

The ACO metaheuristic provides users with an alternative way of constructing longest-path solutions in acyclic (unidirectional) network topologies. Despite the seemingly iterative approach of the ACO method, the method utilizes intelligent selection procedures to perform the optimization and arrive at the longest paths in a prescribed network topology. Past sample case studies indicate quick convergence to the final solution and thus low computational time, and the performance of the applied optimization model can be further increased by varying the topology parameters.

Ongoing work on the subject matter addresses the inclusion of resource-based scheduling techniques to account for AND/OR resource-combination requirements at the network nodes, and ways to generate total-float values for each activity (the current ACO method does not generate these values).

REFERENCES

Christodoulou, S. (2005) Ant Colony Optimization in Construction Scheduling. *In*: *ASCE's 2005 International Conference On Computing in Civil Engineering*, 12-15 July 2005, Cancun, Mexico. American Society of Civil Engineers (ASCE).

Dorigo, M. (1991) Ant Colony Optimization. *In*: G. C. Onwubolu and B. V. Babu (eds.) *New Optimization Techniques in Engineering*. Springer-Verlag Berlin Heidelberg, 101-117.

Dorigo, M. (1992) *Optimization, Learning and Natural Algorithms*, Ph.D.Thesis, Politecnico di Milano, Italy.

Dorigo, M., Maniezzo, V., and Colorni, A. (1996) The Ant System: Optimization by a Colony of Cooperating Agents. *IEEE Transactions on Systems, Man, and Cybernetics Part B*, 26(2), 29-41.

Dorigo, M., Di Caro, G. and Gambardella, L. M. (1999) Ant Algorithms for Discrete Optimization. *Artificial Life*, 5(2), 137-172.

Maniezzo V, Gambardella L.M., De Luigi F. (2004). *In*: G. C. Onwubolu and B. V. Babu (eds.) *New Optimization Techniques in Engineering*. Springer-Verlag Berlin Heidelberg, 101-117.

Stützle, T. and Dorigo, M. (2002). The Ant Colony Optimization Metaheuristic: Algorithms, Applications, and Advances. *In*: F. Glover and G. Kochenberger (eds.) *Handbook of Metaheuristics*. Kluwer Academic Publishers, Norwell, MA.