Efficient Routing of Mobile Agents in a Stochastic Network

Amir Elalouf, Eugene Levner, and T.C.E. Cheng

***Abstract*—Mobile agents are autonomous programs that may be dispatched through computer networks. Using a mobile agent is a potentially efficient method to perform transactions and retrieve information in networks. Unknown congestion in a network causes uncertainty in the routing times of mobile agents so the routing of mobile agents cannot rely solely on the average travel time. In this paper we deal with a given stochastic network in which the mobile agent routing time is a random variable. Given pre-specified values R and PR, the objective is to find the path with the minimum expected time under the constraint that the probability that the path time is less than R is at least PR. We show that this problem is NP-hard, and construct an exact pseudo-polynomial algorithm and an ε-approximation algorithm (FPTAS) for the problem.**

***Index Terms*—Agent-based architecture, fast routing algorithm, FPTAS, stochastic routing.**

I. Introduction

In the context of distributed computer communication networks, we define an *agent* as "a human or software system that communicates and cooperates with other human or software systems to solve a complex problem that is beyond the capability of each individual system" [1]. This definition is compatible with the definitions given by Jennings and Wooldridge [2], Shen et al. [3–5], and Peng et al. [6]. An *autonomous agent-based system* is a system that is able to function in some environments without the direct intervention of human beings or other agents and that has control over its own actions and internal states. Its major advantage is that it can effectively access distributed resources in a low-bandwidth network. In particular, such a system may be useful in a client/server model, in which a client needs to access a huge database on a server.

This access requires a large amount of data to be transmitted over the network and may significantly waste bandwidth. By sending a mobile program to the server and performing data processing at the server, unnecessary data transmission can be avoided. Even if the client/server connection fails, the mobile program can successfully perform its mission.

Mobile agent-based technologies have been used in distributed computer networks for more than two decades. Establishing the notion of mobile agents in 1994, White [7] describes a computational environment known as "Telescript" in which running programs are able to transport themselves from host to host in a computer network. Tsichritzis [8] introduces the notion of mobile computation by describing a hypothetical computing environment in which all the objects are mobile. Within the scope of this paper, we follow the definitions in [3–11] and define a *mobile agent* as "a software agent that is able to autonomously migrate from one host to another in a computer network."

The latest achievements in multi-agent systems have brought new possibilities for integrated systems management. In typical applications, a mobile agent visits several hosts in a network in order to complete its mission. The hosts provide the agent with information and access to services, as well as a platform for carrying out various actions and for communicating with other agents. The services and information that the agent needs to access are distributed across different sites and are available in different forms and at different levels of accuracy and degrees of reliability. This gives rise to a mobile agent routing problem with uncertain data, in which limited computational resources are available at many possible sites.

A given benefit function determines how much benefit (e.g., information from sites, retrieval data, etc.) each site contributes to an agent's mission. Since many different sites provide information yielding different degrees of benefit, the mobile agent should find a best possible itinerary to visit them under resource constraints. The problem of enhancing the efficiency of mobile agents then reduces to the problem of finding resource-constrained extremal paths in a graph. The agent's routing problem consists in finding an information- or resource-constrained route that provides the best agent performance.

In this study we deal with mobile agent routing in a stochastic network. An agent takes an instruction to move from one location to another until it reaches its destination, where each action/move involves uncertainty. The common objective function for an agent is the minimum expected

Manuscript received on June 1, 2012; accepted for publication on August 23, 2012.

A. Elalouf is with Bar-Ilan University, Ramat Gan, Israel (e-mail: amir.elalouf@biu.ac.il).

E. Levner is with Ashkelon Academic College, Ashkelon, Israel (e-mail: elevner@acad.ash-college.ac.il).

T.C.E. Cheng is with Hong Kong Polytechnic University, Kowloon, Hong Kong (e-mail: edwin.cheng@inet.polyu.edu.hk).

performance time or the minimum expected cost. In contrast to many earlier known agent routing problems (see, e.g. [12–15]), we study uncertainty by explicitly taking into account variance in agents' routes and probabilistic path characteristics.

We do this by incorporating a corresponding non-linear constraint into the problem formulation. To the best of our knowledge, there is no work in the literature with a focus on the design of efficient (polynomial-time) solution methods for the constrained stochastic agent-routing problem. Aiming to fill this research gap, we develop a fast ε-approximation algorithm for solving the considered problem. Another contribution of this paper is that, whereas many previous works (e.g., [12–14]) have considered acyclic networks, we allow the network to contain cycles, which makes the problem much more practical. While the simple deterministic shortest path problem can be solved in polynomial time, the considered stochastic agent-routing problem turns out to be NP-hard.

In the next section we provide a brief overview of other works related to our study. In Section III we formulate the problem. In Section IV we present the exact dynamic programming (DP) solution algorithm. In Section V we construct a new fully polynomial time approximation scheme (FPTAS) algorithm. Section VI concludes the paper.

II. Related Works

The following basic mobile agent-routing problems have been studied in the literature:

Problem P1. To maximize the total benefit generated from agent travel, subject to the condition that the total travel time (sometimes called "delay") does not exceed a given threshold. Such a problem has been studied by Camponogara and Shima [12] and by Elalouf and Levner [1].

Problem P2. To minimize the agent's total expected travel time to complete the task under the constraint that the total travel cost does not exceed a given budget limit. Such a problem has been investigated by, e.g., Brewington et al. [15], Hassin [14], Goel et al. [16], and Xue et al. [17], among many others.

For the agent routing task, a computational scheme considering multiple objectives has been pursued by Wu et al. [18], who combine three objectives (communication cost, path loss, and detected signal energy level) into a single function and optimize it using a genetic algorithm that outperforms local heuristics.

To evaluate the effectiveness of multi-objective algorithms against a single-objective approach, Rajoopalan [19] applies a more general weighted genetic algorithm (WGA) iterated with different weights in order to obtain different non-dominated routing solutions.

Osman et al. [20] analyze an execution model for agent routing to develop a pragmatic framework for fault tolerance in agent systems. This framework adopts a communication-pair, independent-check pointing strategy.

In this paper we consider the mobile agent framework described by Rech et al. [21] and by Camponogara and Shima [12]. Specifically, we develop a graph-theoretic model for computing the agent's itinerary under resource constraints, and on the basis of this model we design exact DP and approximation solution algorithms.

In what follows, we suggest a general three-stage technique, which follows and extends an earlier computational scheme suggested by Gens and Levner [22, 23] and by Elalouf and Levner [13] for the Knapsack and routing problems, respectively.

The new technique essentially improves on the algorithms proposed by Camponogara and Shima [12] and Hassin [14] for the deterministic constrained routing problems P1 and P2, and also provides a new way to obtain a fast solution for the stochastic routing problem.

III. Problem Formulation

The problem framework is based on a computational network composed of a graph (possibly cyclic), $G = (N, A)$, with a set N of nodes, a set A of arcs, a start node $s = 1$, and a destination node $t = n$, where $|N| = n$ and $|A| = h$. The term t_{ij}, denoting the time to traverse arc (i, j) in G, is a normal random variable characterized by two parameters: the expected time m_{ij} and the variance v_{ij}. The parameters m_{ij} are assumed to be integers. A path p is called *feasible* if the probability that the path time is less than R is at least P_R, where R and P_R are given values. The problem is to find a feasible path with the minimum expected time.

Problem input: $G(N, A)$: a given graph.

For any arc $(i, j) \in A$, two parameters are given: m_{ij}, the expected time; v_{ij}, the variance.

$M(p)$ denotes the expected time to traverse path p; $M(p) = \sum_{(i,j)\in p} m_{ij}$.

$V(p)$ denotes the variance of the time it takes to traverse path p. We assume that all the times t_{ij} are independent random variables, and therefore $V(p) = \sum_{(i,j)\in p} v_{ij}$.

In a mathematical form, the problem is to find a path p such that

$$\min_p \left(M(p)\right)$$

$$s.t$$

$$M(p) + \phi^{-1}(P_R)\sqrt{V(p)} \le R$$

Note that ϕ^{-1} is the inverse form of the standard normal distribution. The meaning of the constraint is evident, i.e., if the constraint is satisfied, the probability that the traverse time will not exceed R is at least P_R.

IV. Exact Solution Algorithm: Dynamic Programming

This section introduces an exact DP solution algorithm. Since m_{ij} are assumed to be integers, DP is a pseudo-polynomial solution algorithm. Its complexity is estimated below.

Let us associate with each path p a pair (M, V), where $M = M(p)$ is the expected time to traverse path p, and, correspondingly, $V = V(p)$ is the variance of the time to traverse p. We deal with sets $S(k)$ of pairs (M, V), arranged in increasing order of the M-values, so that every pair in $S(k)$ corresponds to a path from node s to a node k. In order to restore the path corresponding to a pair (M, V), we define for each pair a predecessor pair and use standard backtracking.

If there are two pairs in $S(k)$, $(M1, V1)$ and $(M2, V2)$ such that $M1 \le M2$ and $V1 \le V2$, then the pair $(M2, V2)$ is called *dominated* and may be discarded. Let UB be an upper bound on the total expected time for the optimal path. For instance, UB can be set to $\sum_{(i,j)\in A} m_{ij}$. The polynomial time DP solution algorithm is as follows:

Algorithm 1. Exact pseudo-polynomial DP solution

1. Input: $G(N, A)$, $|N| = n$, $|A| = h$, $\{(m(i,j), v(i,j) \mid (i,j) \in A\}$, R
2. Output: A constrained path with minimum expected time
3. **Step 1.** [Initialization]
4. Set $S(1) = \{(0, 0)\}$, $S(k) \leftarrow \varnothing$ for $k = 2, \ldots, n$
5. **Step 2.** [Generate $S(2)$ to $S(n)$]
6. Repeat n-1 times
7. for *each arc* $(u, k) \in A$ (*leading from node u to node k*)
8. $W \leftarrow \varnothing$
9. for each pair $(M, V) \in S(u)$ do
10. if $M + m(u,k) + \phi^{-1}(P_R)\sqrt{V + v(u,k)} \le R$ then $W \leftarrow W \cup \{(M+m(u, k), V+v(u, k))\}$
11. endfor
12. $S(k) \leftarrow$ merge($S(k)$, W); during merging eliminate the dominated pairs
13. endfor
14. End Repeat
15. **Step 3**. [Determine optimal solution]
16. find min M in $S(n)$, denote it by *ans*
17. Return *ans* as the optimal time; use backtracking to find optimal path.

Proposition 1.The complexity of the DP solution algorithm (Algorithm 1) is O($hnUB$).

Proof: Since the times are integers and we discard dominated pairs, there are at most UB pairs in sets W and $S(k)$. Furthermore, constructing W in lines 9–11 requires $O(UB)$ elementary operations, because W is constructed from a single $S(k)$. Merging the sorted sets W and $S(k)$ in line 12, as well as discarding all the dominated pairs, is done in linear time (in the number of pairs, which is at most UB).

In Step 2, lines 5–14, we have two nested loops, where the first one begins at line 6 and the second at line 7. These two loops go over all the arcs n–1 times, so in total we have O(hn) iterations of lines 11–13. Thus, the total complexity of Algorithm 1 is O($hnUB$). □

V. Fully Polynomial Time Approximation Scheme

A. General Description of the FPTAS

Our approach to constructing an FPTAS follows the so-called interval partitioning computational scheme. The interval partitioning technique was originally proposed by Sahni [24] for the Knapsack problem and was later improved by Gens and Levner [21], Levner et al. [25], and Elalouf et al. [1]. We suggest a scheme that consists of three main stages:

Stage A: Find a preliminary lower bound LB and an upper bound UB on the optimal path's expected time such that $UB/LB \le n$.

Stage B: Find improved lower and upper bounds such that $UB/LB \le 2$.

Stage C: Partition the interval $[LB, UB]$ into n/ε equal subintervals, delete sufficiently close solutions in the subintervals (taking only one "representative" from each subinterval), and then find an ε-approximation solution using full enumeration of the "representatives".

This technique is similar to that presented by Elalouf et al. [1]. Note, however, that the type of problem treated in the present paper is more practical than that in [1]. First, the problem considered here is of a stochastic nature, so it is described by a non-linear constraint. Second, its underlying graph G is allowed to have cycles. As a result, the algorithm proposed herein has a different complexity compared with that in [1].

B. Stage A: Finding Preliminary Lower and Upper Bounds

We use the following greedy technique: Let $A = \{a_1, a_2, \ldots, a_h\}$ be the set of arcs in $G(N, A)$. Denote graph $G'(N', A')$ with the same set of nodes, i.e., $N' = N$, and the set of arcs $A' \subseteq A$. To define A', we use the notation x_{ai}, a binary variable. If $x_{ai} = 1$ then $a_i \in A'$; otherwise $a_i \notin A'$. We order the arcs in G in non-decreasing order of their expected times, i.e., $m_{[a1]} \le m_{[a2]} \le \cdots \le m_{[ah]}$, and initialize $x_{ai} = 0$ for any $i = 1, \ldots, h$ (i.e., we initialize G' as a graph with no arcs).

Then we sequentially set $x_{[a1]} = 1, x_{[a2]} = 1, \cdots$ and add each arc to the graph until we obtain a path from the source to the destination that satisfies the constraint.

If all $x_{ai} = 1$ but we cannot find such a path, there is no feasible solution for the problem considered. Let x_k be the last variable that is set to 1 in the above procedure. Then we set $m_0 = m_{ak}$. Obviously, the optimal total travel time (denoted by OPT) must lie between m_0 and nm_0. When OPT equals zero, the above greedy procedure in Stage A finds the exact

optimal solution (i.e., a path of zero duration) and Stages B and C are not required.

Proposition 2. The complexity of the FPTAS in Stage A is $O(n^2 \log h)$.

Proof. Sorting the arcs described above is done in $O(h \log h)$. Each check of whether the graph G has a feasible path on a selected set of arcs requires $O(n^2)$ time [26]. The total number of checks is $O(\log h)$ if we use a binary search in the interval $[1, h]$. Thus, the complexity of Stage A is $O(n^2 \log h)$. □

C. Stage B: Finding Improved Bounds

This stage has two building blocks: a test procedure denoted Test(w, ε), and a narrowing procedure denoted BOUNDS, which uses Test(w, ε) as a sub-procedure. The procedure is similar to the testing method described in [1] and [27], with some minor changes that take the stochastic nature of the problem into account.

Test Procedure (Test(w, ε))

Test(w, ε) is a parametric dynamic-programming type algorithm that has the following property: Given positive parameters w and ε, it reports either that the minimum possible expected travel time is $M^* \le w$ or that that $M^* \ge w(1-\varepsilon)$.

Test(w, ε) will be repeatedly applied as a sub-procedure in the algorithm BOUNDS below to narrow the gap between UB and LB until $UB/LB \le 2$.

Associate a pair (M, V) with each path p, where $M = M(p)$ is the path's expected travel time, and, correspondingly, $V = V(p)$ is the variance of the path time. We deal with sets $S(k)$ of pairs (M, V) arranged in increasing order of the M-values so that every pair in $S(k)$ corresponds to a path from the start node s to a node k. As in the DP solution algorithm above, we discard all the dominated pairs in all sets $S(k)$.

If $M2 - M1 \le \delta$, then the pair (M, V) is called δ-*close*. We discard δ-close pairs from set $S(k)$ according to the following procedure:

(a) Let w be a given parameter satisfying $LB \le w \le UB$. For each $S(k)$, partition the interval $[0,w]$ into $\lceil n/\varepsilon \rceil$ equal subintervals of size no greater than $\delta = \varepsilon w/n$.

(b) If, for a given subinterval, there are multiple pairs from S(k) for which the value of M falls into the subinterval, discard all such δ-close pairs, leaving only one representative pair in the subinterval, namely, the pair with the smallest (in this subinterval) V-coordinate.

(c) Any pair (M, V) with $M > w$ (called w-redundant) must be discarded.

The algorithm for Test(w, ε) is as follows:

Algorithm 2. Testing Procedure (Test(w, ε))

1. Input: $G(N, A)$, $|N| = n$, $|A| = h$, $\{(m(i,j), v(i,j) \mid (i,j) \in A\}$, R
2. Input ε, w
3. $\Delta \leftarrow \varepsilon w/n$
4. **Step 1.** [Initialization]
5. Set $S(1) = \{(0, 0)\}$, $S(k) \leftarrow \varnothing$ for $k = 2, \ldots, n$
6. **Step 2.** [Generate $S(1)$ to $S(n)$]
7. Repeat n-1 times
8. for each arc $(u, k) \in A$ (leading from node u to node k)
9. $W \leftarrow \varnothing$
10. for each pair $(M, V) \in S(u)$ do
11. if $M + m(u,k) + \phi^{-1}(P_R)\sqrt{V + v(u,k)} \le R$ then $W \leftarrow W \cup \{(M+m(u, k), V+v(u, k))\}$
12. endfor
13. $S(k) \leftarrow$ merge($S(k)$, W); during merging eliminate the dominated pairs and the δ-close pairs
14. endfor
15. End Repeat
16. **Step 3.** Find a pair (M, V) in $S(n)$, such that $M \le w$.
17. If such a path is found in $S(n)$, return $M^* \le w$.
18. If such a path cannot be found in $S(n)$ return $M^* \ge w(1-\varepsilon)$

Proposition 3. The complexity of Test(w, ε) is $O(hn^2/\varepsilon)$,

Proof. Since the subinterval length is $\delta = \varepsilon w/n$, we have $O(n/\varepsilon)$ subintervals in the interval $[0, w]$. Therefore there are $O(n/\varepsilon)$ representative pairs in sets W and $S(k)$. Further, constructing each W in lines 10-12 requires $O(n/\varepsilon)$ elementary operations. Merging the sorted sets W and $S(k)$ in line 13, as well as discarding all the dominated pairs, is done in linear time (in the number of pairs, which is $O(n/\varepsilon)$). Step 2 (starting in line 6) goes over all the arcs n-1 times, so in total we have $O(nh)$ iterations of lines 10-12. Thus, the total complexity of Algorithm 2 is $O(hn^2/\varepsilon)$. □

The Narrowing Procedure BOUNDS

The narrowing procedure presented in this section (BOUNDS) is adapted from the procedure suggested by Ergun et al. [27] for solving the restricted shortest path. Specifically, when we run Test(w, ε), we choose ε as a function of UB/LB, updating its value from iteration to iteration. To distinguish the allowable error (ε) in the FPTAS from the iteratively changing error in the testing procedure, we denote the latter as θ. The algorithm proceeds as follows:

Algorithm 3. BOUNDS

1. Input: LB and UB such that $UB/LB \le n$.
2. Output: LB and UB such that $UB/LB \le 2$
3. If $UB/LB \le 2$, Goto 10
4. Set $\theta \leftarrow \sqrt{UB/LB} - 1$
5. Set $w \leftarrow \sqrt{LB \cdot UB/(1-\theta)}$
6. Run Test(w, θ)
7. If Test(w, θ) returns that $M^* \le w$ then set $UB \leftarrow w$
8. else set $UB \leftarrow w(1-\theta)$
9. Go to line 3
10. Return the improved LB and UB
11. End

The complexity of BOUNDS is O(hn^2). The proof is along the same line as that of Lemma 5 in [27].

D. Stage C: The ε-Approximation Algorithm (AA)

We start Stage C with LB and UB values satisfying $UB/LB \le 2$, and obtain an ε-approximation path.

Associate with each path p a pair (M, V), where, as above, $M = M(p)$ is the path expected time, and, correspondingly, $V = V(p)$ is the path variance. We deal with sets $S(k)$ of pairs (M, V) arranged in increasing order of the M-values so that every pair in $S(k)$ corresponds to a path from the start node s to a node k. As in DP, we delete all the dominated pairs in all the $S(k)$ sets. In addition to deleting the dominated pairs, we delete δ-close pairs as follows:

(a) In each $S(k)$, partition the interval [0, UB] into $\lceil (UB/LB)(n/\varepsilon) \rceil$ equal subintervals of size no greater than $\delta = \varepsilon LB/n$;

(b) If, for a given subinterval, there are multiple pairs from S(k) for which the value of M falls into the subinterval, discard all such δ-close pairs, leaving only one representative pair in the subinterval, namely, the pair with the smallest (in this subinterval) V-coordinate.

(c) A pair (M, V) with $M > UB$ may be discarded.

The corresponding algorithm proceeds as follows:

Algorithm 4. ε-approximation algorithm (AA (LB, UB, ε))

1. Input: $G(N, A)$, $|N| = n$, $|A| = h$, $\{(m(i,j), v(i,j) \mid (i,j) \in A\}$, R
2. Input UB, LB, ε
3. $\Delta \leftarrow \varepsilon LB/n$
4. Output: ε-approximation path such that path expected time is at most (1+ ε)OPT
5. **Step 1.** [Initialization]
6. Set $S(1) = \{(0, 0)\}$, $S(k) \leftarrow \varnothing$ for $k = 2, \ldots, n$
7. **Step 2.** [Generate $S(2)$ to $S(n)$]
8. Repeat n-1 times
9. for each arc $(u, k) \in A$ (leading from node u to node k)
10. $W \leftarrow \varnothing$
11. for each pair $(M, V) \in S(u)$ do
12. if $M + m(u,k) + \phi^{-1}(P_R)\sqrt{V + v(u,k)} \le R$ then $W \leftarrow W \cup \{(M+m(u, k), V+v(u, k))\}$
13. endfor
14. $S(k) \leftarrow$ merge($S(k)$, W); during merging eliminate the dominated pairs and the δ-close pairs
15. endfor
16. End Repeat
17. **Step 3**. [Determine approximate solution]
18. find min M in $S(n)$, denote it by *ans*
19. Return *ans* as the ε-approximation expected time, use backtracking to find the path
20. The path's expected time is at most (1+ε)OPT.

Theorem 1. The complexity of AA(LB, UB, ε) is O(hn^2/ε). The complexity of the entire three-stage FPTAS is O(hn^2/ε).

Proof: Since the subinterval length is $\delta = \varepsilon LB/n$, we have $O(n(UB/LB)(1/\varepsilon))$ subintervals in interval [0, UB], and since $UB/LB \le 2$, there are O(n/ε) subintervals in the interval [LB, UB]. Therefore, there are O(n/ε) representative pairs in any set W, T, and $S(k)$.

Constructing each W in lines 11–13 requires O(n/ε) elementary operations because W is constructed from a single $S(k)$. Merging the sorted sets W and T in line 14, as well as discarding all the dominated pairs, is done in linear time (in the number of pairs, which is O(n/ε)). In Step 2 we have O(nh) iterations of lines 11–13. Thus, the total complexity of Algorithm 4 is O(hn^2/ε). Since Step C dominates Steps A and B of the algorithm, the complexity of the entire approximation algorithm is O(hn^2/ε). □

VI. Concluding Remarks

The main contribution of this work is a novel routing scheme for mobile agents in a wireless stochastic network that optimizes agent performance and reduces possible delays. An auxiliary dynamic programming algorithm running in pseudo-polynomial time is proposed for developing a fast routing strategy.

Notably, algorithm complexity is thoroughly analyzed. The mathematical model and algorithms presented in this paper can serve as a prototype for future commercial protocols for mobile agent routing over stochastic networks.

Future research should focus on developing more realistic models and solution algorithms that incorporate a broader variety of the practical characteristics of real-world computer and communication networks.

References

[1] A. Elalouf, E. Levner, and T. C. E. Cheng, "Efficient routing of mobile agents for agent-based integrated enterprise management: A general acceleration technique," *Lecture Notes in Business Information Processing*, vol. 88, pp. 1–20, 2011.

[2] N. R. Jennings and M. J. Wooldridge, "Applications of Intelligent Agents," in *Agent Technology: Foundations, Applications, and Markets*, N. R. Jennings, M. J. Wooldridge, Eds., Heidelberg: Springer, 1998, pp. 3–28.

[3] W. Shen, D. H. Norrie, and J.-P. Barthes, *Multi-Agent Systems for Concurrent Intelligent Design and Manufacturing*. London: Taylor and Francis, 2001.

[4] W. Shen, D. Xue, and D. H. Norrie, "An agent-based manufacturing enterprise infrastructure for distributed integrated intelligent manufacturing systems," in *Proceedings of the Third International Conference on the Practical Application of Intelligent Agents and Multi-Agents*, London, UK, 1997, pp. 1–16.

[5] W. Shen, "Distributed manufacturing scheduling using intelligent agents," *IEEE Intelligent Systems*, vol. 17, pp. 88–94, 2002.

[6] Y. Peng, T. Finin, Y. Labrou, B. Chu, J. Long, X. Tolone, and A. Boughannam, "A multi-agent system for enterprise integration," in *Proc. of PAAM'98*, London, UK, 1998, pp. 155–169.

[7] J. E. White, *Telescript Technology: The Foundation for the Electronic Marketplace*, White Paper, Mountain View, CA, USA: General Magic, Inc., 1994.

[8] D. Tsichritzis, *Objectworld, Office Automation*. Heidelberg: Springer-Verlag, 1985.

[9] W. Shen, Q. Hao, H. J. Yoon, and D. H. Norrie, "Applications of agent-based systems in intelligent manufacturing: An updated review," *Advanced Engineering Informatics*, vol. 20, pp. 415–431, 2006.

[10] T. Papaioannou, *Using Mobile Agents to Improve the Alignment between Manufacturing and Its IT Support Systems, Robotics and Autonomous Systems*. Amsterdam: Elsevier, 1999.

[11] W. Shen, F. Maturana, and D. H. Norrie, "MetaMorph II: An agent-based architecture for distributed intelligent design and manufacturing," *Journal of Intelligent Manufacturing*, vol. 11, pp. 237–251, 2000.

[12] E. Camponogara and R. B. Shima, "Mobile agent routing with time constraints: A resource constrained longest-path approach," *Journal of Universal Computer Science*, vol. 16, pp. 372–401, 2010.

[13] A. Elalouf and E. Levner, "General techniques for accelerating FPTAS for the routing and knapsack problems," in *Abstract Book, Annual Meeting 2011 of Operations Research Society of Israel (ORSIS 2011)*, Akko, Israel, 2011, p. 14.

[14] R. Hassin, "Approximation schemes for the restricted shortest path problem," *Mathematics of Operations Research*, vol. 17, 36–42, 1992.

[15] B. Brewington, R. Gray, K. Moizumi, D. Kotz, G. Cybenko, and D. Rus, "Mobile agents in distributed information retrieval," in *Intelligent Information Agents*, M. Klusch, Ed., Heidelberg: Springer Verlag, 1999, pp. 355–395.

[16] A. Goel, K. G. Ramakrishnan, D. Kataria, and D. Logothetis, "Efficient computation of delay-sensitive routes from one source to all destinations," in *IEEE Infocom'2001*, Washington, DC: IEEE Press, 2001, pp. 854–858.

[17] G. Xue, A. Sen, W. Zhang, J. Tang, and K. Thulasiraman, "Finding a path subject to many additive QoS constraints," *IEEE Transactions on Networking*, vol. 15, pp. 201–211, 2007.

[18] Q. Wu, N. S. V. Rao, J. Barhen, S. S. Iyengar, V. K. Vaishnavi, H. Qi, and K. Chakrabarty, "On computing mobile agent routes for data fusion in distributed sensor networks," *IEEE Trans. Knowledge and Data Engineering*, vol. 16, pp. 740–753, June 2004.

[19] R. Rajagopalan, C. K. Mohan, P. Varshney, and K. Mehrotra, "Multi-objective mobile agent routing in wireless sensor networks," in *Evolutionary Computation, 2005. The 2005 IEEE Congress on 2–5 Sept. 2005*, vol. 2, 2005, pp. 1730–1737.

[20] T. Osman, W. Wagealla, and A. Bargiela, "An approach to rollback recovery of collaborating mobile agents," IEEE Trans. Systems, Man and Cybernetics, Part C, vol. 34, pp. 48–57, Feb 2004.

[21] L. Rech, R. S. Oliveira, and C. B. Montez, "Dynamic determination of the itinerary of mobile agents with timing constraints," in *Proc. IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, Compiegne, France, 2005, pp. 45–50.

[22] G. V. Gens and E. V. Levner, "Fast approximation algorithms for job sequencing with deadlines," *Discrete Applied Mathematics*, vol. 3, pp. 313–318, 1981.

[23] G. V. Gens and E. V. Levner, "Fast approximation algorithms for knapsack type problems," in *Lecture Notes in Control and Information Sciences*, vol. 23, Berlin: Springer Verlag, 1980.

[24] S. Sahni, "Algorithms for scheduling independent tasks," *Journal of the ACM*, vol. 23, pp .116–127, 1976.

[25] E. Levner, A. Elalouf, and T. C. E. Cheng, "An improved FPTAS for mobile agent routing with time constraints," *Journal of Universal Computer Science*, vol. 17, 1854–1862, 2011.

[26] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. Cambridge, MA: MIT Press, 2001.

[27] F. Ergun, R. Sinha, and L. Zhang, "An improved FPTAS for restricted shortest path," *Information Processing Letters*, vol. 83, pp. 287–291, 2002.