Deciding About Agent Mobility Using a Cost Performance Model

Dalia Elmansy, Ahmed Sameh
Dept. of Computer Sc.,The American University in Cairo,
P.O.Box 2511, Cairo, Egypt

Abstract Mobile Agents can accomplish a task through different means of communication during their life span. They can either communicate by migrating from one node to another in the network, or by communicating with nodes of interest through traditional remote procedure calls. It has been proven that a mobile agent communicates optimally when it mixes between migration and remote communication taking into consideration the specific characteristics of the application, and it is finding the proper mix of migration and remote communication, that is the real challenge that can impact the overall mobile agent performance. This challenge is, in fact, an optimization problem with the goal of finding the optimal agent migration sequence, and hence the optimal agent communication strategy. This paper extends and uses previous research conducted on mobile agents regarding their communication performance models, with the focus on developing an efficient agent migration and communication model, which in turn will enhance the overall mobile agent's performance. This is achieved by providing the mobile agent with efficient, optimal, adaptable algorithms to solve the problem of finding the critical sequence of mixed agent migration and remote procedure calls, under certain restrictions on the sequence of interactions of the mobile agent.

Key-Words: Mobile Agents, RPC, Performance Model, Dijkstra, A*, Priority Queues.

1. Introduction

Mobile agents communication can be carried out in different forms. The two extreme forms of communication is pure RPC communication, or communication through agent migration. In the later case, the agent is always mobile. As such, the agent would accomplish its task through migrating to different nodes in the network one after the other, and sends its reply back to its starting node. In the earlier case, the agent is always stationary. As such, the agent would stay in its location and would accomplish its task through sending messages to other network nodes of interest. In between these two extremes, the agent could be strategically mobile. In such approach the agent accomplishes its task by migrating to some nodes in the network, and by sending messages to some other nodes [Chia97].

In [Chia97] it was mentioned that mobile agents have many advantages, which basically include overall network traffic reduction, reduction in connectivity requirements and enabling control of remote operations with real-time constraints. However, to fully exploit the advantages that mobile agents provide, they must be designed in the context of the application domain. Thus they can utilize knowledge about the problem to be solved, in order to make optimal mobility decisions or strategies, which naturally translates to optimal use of computing and network resources.

So a comparison between the three cases mentioned earlier was carried out in [Chia97]. It was found that a strategically mobile agent, or a mobile agent that extracts relevant characteristics of the application (or even that is aware of a subset of the characterizing parameters of the application) uses the network most sparingly in terms of total transferred bytes. This is illustrated in ***figure 1*** below.

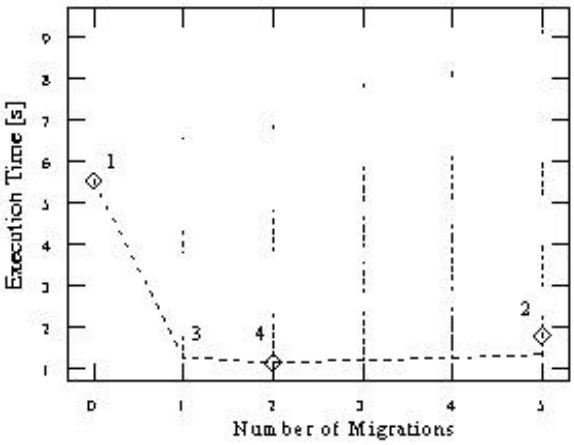


Figure 1 - Performance of Mobile Agent's Different Communication Choices [Chia97]

In ***figure 1*** execution time goes to minimum when communication is carried out as a certain mix of migration and RPC, i.e.
Figure 1 - Performance of Mobile Agent's Different Communication Choices [Chia97]
when the agent is neither "Always Stationary" (where number of migrations=0) nor "Always Mobile" (where number of migrations is maximum).

This directs our attention to the fact that mobile agent technology must be integrated with non-mobile architectures (e.g. client-server, peer-to-peer), sometimes, in order to realize its promise of providing scalable and optimal use of network resources. As such, the major challenge for wider use of mobile agents is the proper integration of this technology with non-mobile concepts, while taking into account the specific needs of applications. Thus, the proper decision strategy as of when to migrate and when to communicate through messages would be the real challenge that can greatly impact the mobile agents overall performance.

2. Problem Statement

A mobile agent communicates optimally when it combines between migration and RPC, while taking into account the specific needs of the application. It is finding the proper combination that is the real challenge that can greatly impact the overall mobile agents' performance. This is an optimization problem to find the optimal agent migration sequence, which should be adaptable to network changes, in order to support mobile and partially-connected computing.

3. An Enhanced Performance Model for Mobile Agents Interaction

In [Stras97] a general performance model for interaction between agents in mobile agent systems was introduced. The two interaction models considered were remote procedure call **(RPC)** and mobile agents **(MA)**, since remote evaluation **(REV)**, in the context of mobile agents, was found to be similar to **MA** in terms of their communication needs, i.e. it involves only the transport of agent code and some parameters.

The RPC and migration performance model calculates the cost of making a single RPC and a single agent migration from one location in the network to another. This cost is measured in terms of network load and execution time. However, this model assumes a uniform network with homogeneous nodes; i.e. all the nodes in the network have the same relative speed. This means that in order to support mobile and partially-connected computing, where processing power of nodes vary, the model has to be modified to work on a heterogeneous network, with various types of nodes having different relative speeds.

3.1. Interaction by RPC

Given a *uniform network*, with load B_{RPC} (in bytes), a simple RPC made from location ***L1*** to location ***L2*** would consist of the size of the request B_{req} and the size of the reply B_{rep}, thus:

$$B_{RPC}(L_1, L_2, B_{req}, B_{rep}) = \begin{cases} 0 & \text{if } L_1 = L_2 \\ B_{req} + B_{rep} & \text{else} \end{cases}$$

[Stras97]

The execution time T_{RPC} for a simple RPC from location ***L1*** to location ***L2*** consists of the time for marshalling and unmarshalling

of request and reply plus the time for transfer of the data on the network with delay $\delta(L1, L2)$ and throughput $\tau(L1, L2)$. Marshalling and unmarshalling time should depend on the nodes processing power as well as the size of the request and reply parameters.

S0 $T_{marshalling} = (B_{req} + B_{rep})/RS(L_1) + (B_{req} + B_{rep})/RS(L_2)$ ♦

Where $RS(L)$ represent the relative speed of the node at location L

And $T_{RPC}(L_1, L_2, B_{req}, B_{rep}) = 2\delta(L_1, L_2) + 1/\tau(L_1, L_2) * B_{RPC}(L_1, L_2, B_{req}, B_{rep}) + T_{marshalling}$

3.2. Interaction by Agent Migration

If the agent consists of B_{code} bytes of code, B_{data} bytes of data and B_{state} bytes of execution state and is described by $B_A = (B_{code}, B_{data}, B_{state})$

So the network load B_{mig} resulting from the migration of an agent A from location $L1$ to location $L2$ is calculated by:

$$B_{Mig}(L_1, L_2, B_A) = \begin{cases} 0 & \text{if } L_1 = L_2 \\ P(B_{cr} + B_{code}) + B_{data} + B_{state} & \text{else} \end{cases}$$

[Stras97]

Where P denotes the probability that the code is not yet available at location $L2$ and B_{cr} is the size of the request from location $L2$ to location $L1$ to transfer the code. And in order to support mobile/partially connected computing, the marshalling and unmarshalling time would be a function of B_{data} and B_{state} as well as the relative speed RS of the communicating nodes.

Thus $T_{marshalling} = (B_{data} + B_{state})/RS(L_1) + (B_{data} + B_{state})/RS(L_2)$ ♦

Where $RS(L)$ represent the relative speed of the node at location L.

And $T_{Mig}(L_1, L_2, B_A) = (1 + 2P)2\delta(L_1, L_2) + 1/\tau(L_1, L_2) * B_{Mig}(L_1, L_2, B_A) + T_{marshalling}$

If $L \neq L_2$

Else $T_{Mig}(L_1, L_2, B_A) = (1 + 2P)2\delta(L_1, L_2) + 1/\tau(L_1, L_2) * B_{Mig}(L_1, L_2, B_A)$

Now given the above performance models for RPC and agent migration, and given a mobile agent that needs to accomplish a certain task, and the communication sequence that the agent must take in order to finish the task. The issue is how to find the best migration sequence, such that the overall cost of the task is minimal, hence, the overall mobile agent system performance is best. This was transformed to an optimization problem in [Iqbal98] using the unmodified RPC and agent migration performance models.

4. The Problem of Finding the Optimal Agent Migration Sequence

In [Iqbal98], the problem of Finding the Optimal Agent Communication Strategy was transformed into an optimization problem with the goal of finding the shortest agent migration path, and was represented by an optimal decision graph of size n by z, where n denotes the number of communication steps and z denotes the number of migration nodes. A tentative solution algorithm was presented to find the shortest migration path. However, there are some concerns about that solution. Although it is $O(nz^2)$, yet it is inefficient (as it expands all the nodes in the graph whether needed or not), it is neither adaptable to changes in network parameters, nor to node disconnection or failure. Also it doesn't support mobile and partially connected computing, which is one of the domains for which mobile agents paradigm is very promising. On the contrary, it uses the performance cost model proposed in [Stras97] as is, assuming a uniform network, and ignoring the relative speed of nodes, which is never the case in a real network situation, especially if mobile computing is involved. Also, the algorithm presented in

♦ This is the enhanced RPC model; in order to account for mobile/partially connected computing

♦ This is the enhanced agent migration model, in order to support mobile/partially connected computing

[Iqbal98] is not really well formed, its optimality is not proved, and its performance is not measured empirically.

5. The Presented Solution

Given the proposed enhanced performance model for RPC and migration (the model that counts for mobile computing), and given a mobile agent that needs to accomplish a certain task, and the communication sequence that the agent must take in order to finish the task, the issue is how to find the best migration sequence, such that the overall cost of the agent task (in terms of execution time) is minimal, hence, the overall mobile agent system performance is optimal. Such solution should consider the case when communication sequence of the agent is not fixed, like in the case of conditional calls. Also, the proposed approach should allow for the adaptability of the solution to any changes in the network parameters (e.g. any change in network delay or throughput which results in a change in communication costs) or any node failure, overload or disconnection.

That is to say we need to apply an efficient optimal search algorithm such that we get the optimal agent migration sequence, while guarantee a reasonable worst case (let it be $\mathbf{O(}nz^2\mathbf{)}$).

We need to make such algorithm work when the agent's communication sequence is variable, we need to make it adaptable as well to changes in network parameters and to nodes failure and disconnection. Such algorithm could be parameterized, as well, to adjust the tradeoff between the agent's planning cost (i.e. search cost) and the solution quality. This can be done by gray-scaling the cost, through giving the user, for example, the facility to group migration nodes into groups of nodes based on their location and relative speed. This is based on the assumption that nodes in proximity, like within a LAN, and nodes with similar processing power, would have relatively the same migration cost and RPC cost.

Choices of Shortest Path Algorithms: Dijkstra and A Algorithms*

There are many choices for the shortest path algorithm that we can use for the application of finding the optimal agent migration path problem. One of the obvious choices is the most famous single source, shortest path algorithm by Dijkstra [Dijkstra59], which is known to be an efficient optimal algorithm.

Why Dijkstra's ?

We chose Dijkstra's algorithm because it is efficient (not all of the nodes must be expanded in order to reach for the goal), and because it is the best-known algorithm for the problem in theory, and the most robust in practice [Dijkstra59] [Drey68] [Gold96].

Dijkstra's algorithm, is generally, a greedy, uninformed search algorithm that searches the whole search space in order to reach for the optimal solution. Generally, the complexity of the straightforward implementation of the algorithm is known to be quadratic. However, implementations of the Dijkstra's Shortest path algorithm, using Priority Queues built on binary, k-nary or Fibonacci heaps, result in lower complexity for a fully connected graph with $\boldsymbol{n}$ vertices an $\boldsymbol{m}$ arcs. It was proven that Dijkstra's shortest path algorithm that uses Priority Queues based on binary heaps performs even better than ones based on Fibonacci heaps [Gold96].

Why A?*

A* algorithm starts out going straight for the goal. It looks at the path that has the lowest cost from the beginning to the end, and uses a heuristic function to guide it through the way. That's why A* although greedy like Dijkstra's yet is an informed optimal search algorithm, whose performance depends very much on the accuracy of the evaluation (heuristic) function it uses. A* however doesn't change the complexity of our problem, as compared to Dijkstra's, yet it can improve the performance of the average cases, especially if the heuristic function is accurate [Russel95].

6. Problem Definition

Given a fixed communication sequence in terms of $\boldsymbol{CS(i)}$, $1 \leq \boldsymbol{i} \leq \boldsymbol{n}$, the migration cost matrix **MCost(a, b)**, the RPC matrix **RCost(c,d)**, find a migration sequence $\boldsymbol{MS}$ so that **TCost** is minimal.

The enhanced cost calculation function for each node is:

$$\mathbf{Cost}(i) = \mathbf{MCost}(MS(i-1), MS(i)) + \mathbf{Rcost}(MS(i), CS(i))$$

The total cost of communication would be $\mathbf{TCost} = \sum_{i=1 \text{ to } n} \mathbf{Cost}(i)$

Where MCost and RCost are calculated from the enhanced performance model, see sections 3.1, and 3.2.

7. Mathematical Formulation

Objective *Minimize*:

$$\mathbf{TCost} = \sum_{i=1 \text{ to } n} \mathbf{MCost}(MS(i\text{-}1), MS(i)) + \mathbf{RCost}(MS(i), CS(i))$$

Subject to:

Constraints (n constraints) $1 \le MS(i) \le \mathbf{Z}$ Where $1 \le i \le n$

Free Variables $MS(1)..MS(n)$ at each step of the n steps, we have a free variable $MS(i)$, which stands for a decision of choosing a node among the **Z** nodes at each communication step.

Coefficients are all ones.

8. The Algorithms to Find the Shortest Migration Sequence – *Dijkstra* and *A**

8.1. Dijkstra's Algorithm

Dijkstra's single source shortest path algorithm, starts by expanding the starting node **S** and it adds its children to a set of *Reachable Nodes.* Next, every node in the set of *Reachable Nodes* is evaluated based on its cost (see cost equation in section 6). Then all the nodes reachable from the start node and in the set *Reachable Nodes*, are added to the set of *Terminal Nodes*. The set *Terminal Nodes* contains all the nodes that have a final distance to the start node **S**, in the first step, it will only contain the set of nodes directly reachable from **S**. Then the node of minimum cost is chosen from the set *Terminal Nodes* and is expanded to its children. Every time a minimum cost node is chosen from *Terminal Nodes* and is expanded, it is replaced in the set with its children. The algorithm keeps repeating the previous step, until the next node to be expanded is the end node **E**. The shortest path would be the sum of shortest edges from **S** to **E**, and it is traced back from **E** to **S**, as the sum of the edges leading to **S** with minimum total cost.

8.2. A* Algorithm

The A* algorithm uses the following evaluation function to guide the search towards the end node:

$$f(i,j) = g(i,j) + h(i,j)$$

Where $g(i,j)$ is the actual path cost from the start node to reach the node(i,j) , $\mathbf{h(i,j)}$ is the heuristic function estimating the remaining cost from the current location to the goal, and $f(i,j)$ is the estimated cost of the cheapest solution through node (i,j)

In our case :

$$f(i,j) = Nodes(i,j).RCost + Nodes(i,j).h$$

Where $Nodes(i,j).h$ is the heuristic function.

The Heuristic Function:

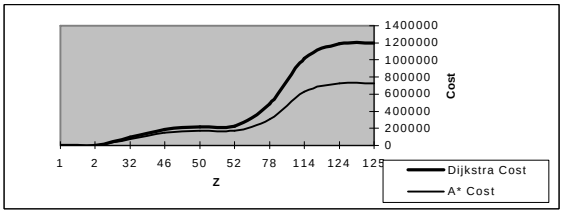
The heuristic function is an estimation of the remaining cost from the current location to the goal. In our problem the cost from the current node location to the goal consists of two parts: the sum of RPC costs of the nodes on the path to the goal, and the sum of migration costs from one step on the path to the other until the goal. Since the heuristic function has to provide an estimate but should never overestimate, and since the lower bound of the migration cost is zero (when the agent does not migrate and communicates from the same node using RPC only) then the optimistic estimation of the migration cost is zero.

The second part is RPC cost. Since the agent has to pass through all the layers in the ODG (Optimal Decision Graph) from the current node to the end node. At each layer the best case is that the agent travels to the cheapest RPC node. Then the most optimistic estimation of the RPC cost is the summation of the cheapest RPC node in each layer down to the end node. Therefore, the heuristic function value is the sum of minimum RPC costs from the current node to the end. It is clear that it is an admissible heuristic, because it is optimistic and ignores the migration cost, thus it thinks that the cost of

solving the problem is less than it actually is. This optimism transfers to the evaluation function ***f(i,j)*** itself, which will thus never overestimate the actual cost of the best solution through ***(i,j)*** [Russel95]. At the same time, the heuristic function is not very inaccurate, because, in its optimism about the cost of the remaining path to the goal, it only ignored the migration cost, which should be always much smaller than the RPC cost for a typical mobile agents application domain.

9. Simulation Results

Both Dijkstra and A* algorithms, are applied on the problem of finding the agent's shortest migration sequence, given a communication sequence. The two algorithms are run on the same set of inputs, for different values of **Z** (Number of the migration nodes) and **N** (Number of communication steps).Since the two algorithms are optimal, and since A* uses a permissible heuristic function, the result was always the shortest migration path. However, the overall cost, in terms of execution time paid by the A* algorithm, proved to be much less than the corresponding cost paid by the Dijkstra's algorithm in order to find the shortest path. ***figures 3*** and***4*** illustrate the gain in cost reduction as a result of using A* for different number of migration nodes (Z) and for different number of communication steps (N).



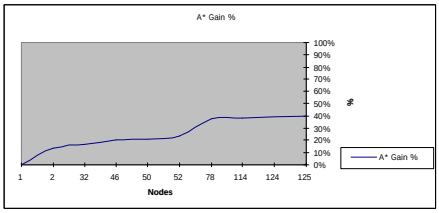


Figure 3 - Costs of finding the shortest path for different values of Z, N=80 Figure 4 - A* improvement in terms of cost reduction

The grouping of the migration nodes is carried out by running the A* algorithm using different tolerance values•. ***Figures 5*** depicts the effect of increasing grouping tolerance on the size of the migration set for different values of **N**. The bigger the tolerance, the more the grouping and hence, the less the size of the migration set **Z**. ***Figure 6*** further demonstrates the effect of increasing grouping on the cost of the algorithm, where it is shown that the bigger the grouping tolerance value the less the cost of the algorithm. Finally ***figure 7*** highlights the tradeoff between solution cost and quality, as gray-scaled by grouping.

In order to simulate a modem-connected computer, a laptop that connects and disconnects from the network as in the case of mobile and partially connected computing, some random node costs were increased. A node, whose cost increases to an infinite value, represents a node that failed or disconnected. The results were tested on different data sets and different cost changes, and the agent's awareness of mobile and partially connected computing was designated by its dynamic changing of the path when it gets to be non-optimal as a result of the change, and its dynamic choice of the new optimal path, that avoids the failed or disconnected node. ***Table 1*** below illustrates the effect of mobile computing support in the presented solution. The minus sign in **path 2** designates the original path before the change, the plus sign designates the changed path segment as a result of the node going mobile. N= 20, Z=100, Y=5, Node No. 28 went mobile in **Path 2** at **communication step #11**

The adaptability of the shortest path algorithm, was tested by simulating some changes in the network environment like changes in RPC cost of a node, or changes in the migration cost to a node. This would happen if a network node fails, disconnects or gets overloaded, or changes happen in links delays or throughputs. The A* algorithm was run several times for different values of N, every time simulating different

• Tolerance is the input value to the grouping algorithm, to which the difference between nodes is compared. Difference between nodes is measured in terms of the Euclidean distance between them. The nodes with Euclidean distance < Tolerance are grouped together.

cost changes, and the total costs incurred by the adaptable and the non-adaptable algorithms are illustrated in ***figure 8***. The results show that the adaptable algorithm always out-performs its non-adaptable counterpart, that has to repeat the whole process of calculating the shortest path from the start node, should any change in network conditions occur. ***Figure 9*** shows the percentage cost gain as a result of adaptability.

10. Conclusion

With the above promising results, the next step in this research will be to support for real-time constraints in the mobile agent's application domain. So if we have a shopping agent that has a shopping list to buy, every item on this list is to be bought from a different shop, and the sequence of shops are given to the agent beforehand (like in our case so far) but every shop, being in a different country, has a different working hours schedule. Then the problem of the agent would be to optimize the migration path while satisfying such real-time constraints of shops working hours, so that it doesn't have to wait. So based on the work done in this research, the next challenging step is to adapt the presented methodology to support real time constraints.

Other future trends in this research include, support for persistent storage, so that an agent may leave some of its data (e.g. results) at one host, carry a small part of it, and yet be able to remotely access the saved data if necessary. Also, Getting information from network routers, and some hosts about network connectivity and delays.

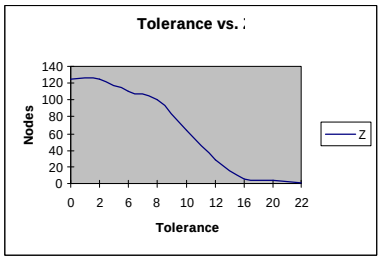


Figure 5 – Effect of grouping tolerance on and the size on the migration set, N=80

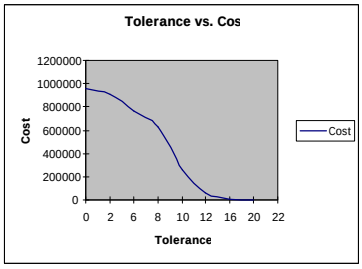


Figure 6: Effect of increasing grouping tolerance; Group size, on algorithm cost, N=80

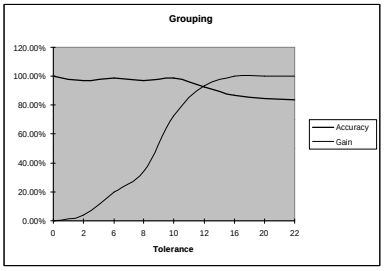


Figure 7: Tradeoff between cost reduction solution quality as a result of grouping, N=80

Comm Step	Path 1	Path 2	Comm Nod	Comm Step	Path 1	Path 2	Comm Node
11	28	28 -	2	1	1	1 -	1
12	28	1 +	2	2	1	1 -	1
13	1	1 +	1	3	1	1 -	2
14	1	1 +	3	4	1	1 -	3
15	1	1 +	1	5	1	1 -	1
16	1	1 +	4	6	1	1 -	1
17	1	1+	1	7	1	1 -	1
18	1	1 +	3	8	1	1 -	3
19	1	1 +	4	9	28	28 -	2
20	1	17 +	1	10	28	28 -	4

Table 1 – Mobile Computing: node #28 disconnection

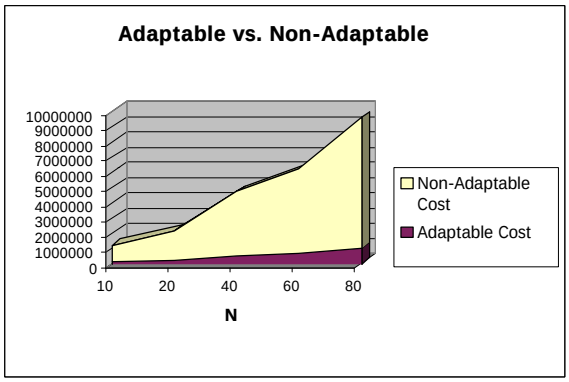


Figure 8 – Adaptable vs. Non-adaptable algorithms

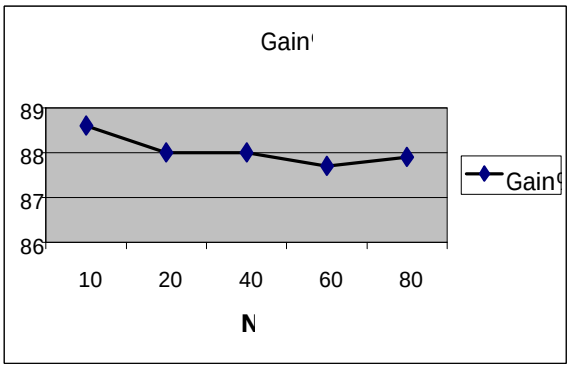


Figure 9 – Percentage cost gain as a result of adaptability

Bibliography

[Chia97] Teck-How Chia, Strikanth Kannapan. *Strategically Mobile Agents*. Proc. 1st Int. Workshop on Mobile Agents, MA97, Springer Verlag, 1997

[Dijkstra59] E. W. Dijkstra. *A Note on Two Problems In Connexion with Graphs*. Numerische Mathematik, 1959

[Drey68] Stuart E. Dreyfus. *An Appraisal of Some Shortest-Path Algorithms*. University of California, Berkeley, California, 1968

[Gold96] Andrew V. Goldberg, Robert E. Tarjan. *Expected Performance of Dijkstra's Shortest Path Algorithm*. NEC Research Institute, 1996

[Iqbal98] A. Iqbal, J. Baumann, M. Strasser, *Efficient Algorithms to Find Optimal Agent Migration Strategies*, University of Stuttgart, IPVR, 1998

[Russel95] Stuart Russel, Peter Norvig. *Artificial Intellgence, A Modern Approach*. Prentice Hall International Editions, 1995

[Stras97] Markus Strasser, Markus Schwehm. *A Performance Model for Mobile Agent Systems*. Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications, 1997