Performance Optimization for Mobile Agent Message Broadcast Model Using V-Agent

Faiz Al-Shrouf, Mohd Eshtay and Khaled Abu Humaidan

Applied Science University
Faculty of IT, P.O.Box 11931, **Amman-Jordan**

Abstract
Mobile agents have motivated the new creation of methodology for parallel distributed computing systems. In this paper, we have developed a model based on Master-Slave Design Pattern of mobile agent systems called Master-V-Slave (MVS) model throughout a computational agent, namely (V-agent). The proposed model utilizes V-agent to analyze a new computing model, namely Message Broadcast Model (MBM) for managing broadcast messages between a set of master agents carrying out tasks on a set of clients and corresponding slave agents receiving and performing tasks within a predefined allocating time on a set of servers. We analyzed the MBM using Vogel's Approximation Method (VAM) to optimize and minimize message time performance distribution architecture. Furthermore, we implemented a prototyping scenario of MBM that computes total number of expected delivery messages from a set of master agents to set of slave agents based on slave time message response with V-agent. Our proposed model could potentially exploit Message Delivery/Response Matrix (MDRM) based on MBM that can be integrated with parallel distributing computing systems and knowingly analyses approaches in operations research. This paper elaborates performance optimization analysis of mobile agents and mathematical computational agents.

Key Words:
Mobile Software Agents, Message Broadcast Model (MBM), Master-V-Slave (MVS) model, Message Delivery/Response Matrix (MDRM), V-Agent.

1. Introduction

Computational architectural design models for distributed systems have been categorized into four design architectures: repository model, client/server model, and layered model [5]. Data is processed between different resources that incorporate multiple machines and numerous computing stages. Drawbacks of design models include performance, security, and flexibility and scalability.

Mobility model [1] [2] is developed as a massive model to overcome aforementioned models shortcomings. Several features were proposed to adhere with mobility agents including: reduce network traffic, their ability to operate asynchronously and autonomously of the process they created them, and assist to construct more robust and fault tolerant systems [8].

Mobile agent patterns [3] are emerging in the design of Agent Oriented Engineering (AOE), and several of these patterns were given intuitive meaningful names such as messenger, notifier, and master-slave pattern. Master-slave pattern is a common task design model incorporated in a broad domain of parallel applications. This model is based on a divides and conquers fashion in which a master delegates tasks to one or more slaves that in turn are distributed throughout the system and work in parallel. Our proposed work is focused to build a new model MVS that utilizes a computational agent (V-agent),which carries out computational algorithm for master delivery messages and slave response time from a proposed computing MBM[1].

This approach of computation is advantageous in that V-agent can be used as a broker, which computes time optimization throughout Vogel's Approximation Method (VAM) [9] and computes total number of messages to be delivered by master agents working on clients to slave agents carrying out tasks on servers. This optimization technique can be exploited, enhancing both the performance and flexibility of mobile agent systems.

2. Related Work

Several authors have presented several techniques for performance optimization analysis of mobile agent systems [6] and mobile agent migration [7]. The developed computing model, called Mobile Agent Parallel Processing Computing (MAPPC) Model using a family of mobile agents. MAPPC uses a Matrix multiplication task and is divided on row wise basis. Based on a number of available servers and dimension of block size is assigned to each mobile agent. The MAPPC model experiments an implementation where small amount of data (matrix of size less than 100) is to be processed on remote servers (less than 3) gives little performance than large matrices on large number of servers, which gives better performance.

MAPPC model uses master-slave pattern for mobile agents in which a slave agent does a given task using CPU resource of remote server. A slave agent the constructs the message and embeds the result into it and sends it to the master agents proxy. The master agent who is waiting for the result from slave agent receives the message and

[1] This research is sponsored by Applied Science University (ASU) and the Faculty of IT.

Manuscript received August 5, 2008
Manuscript revised August 20, 2008

extracts the result and combines all results together. Then it calculates the turnaround time for the computation which will be used to analyze performance.

In our approach, we put our effort to analyze the performance using V-agent optimization model MVS system. V-agent utilizes MBM between master agents who send messages to slave agents who respond and carrying out tasks. Then V-agent calculates the time and number of messages to be delivered for each slave based on the time computing mechanism. Furthermore, V-agent uses VAM approximation that optimizes and minimizes time allocating between master agents and slave agents. This will be used to enhance performance optimization.

3. Motivation

Communication is the base mechanism for coordination and collaboration in mobile agent systems. However current mobile agent systems are not based on the message content, but they focus on three message components message transport, delivery and response, and time allocating, implying difficulties in activity coordination and performance.

Towards this end, several authors are working to enhance message performance mechanism [4] demonstrates an approach for mobile agent communication which can improve performance content filtering system and can unify several message system in a single one.

Our approach addresses message performance optimization through coordination in mobile agent systems using different techniques and mathematical model founded earlier in operations research techniques, such as VAM technique. This approach proved advanced solutions to some basic mathematical models in optimization and reliability system measurements. However, such approaches can be used to improve performance optimization in mobility message delivery, response, and time allocation.

4. Overview Of Mvs Design Model

Master slave pattern [3] is parallel computing application in which master agents working on clients creates slave agents dispatch to remote servers. Slave agents visit specified server to perform the required task. Master agents connect with slave agents using master agents proxy and sends messages in a broadcast paradigm. A user, who wants to perform the task, submits the task to master agents then divide the task into subtask and assign it to individual slave agent. Fig 1, shows the block architecture of master slave task pattern message connection.

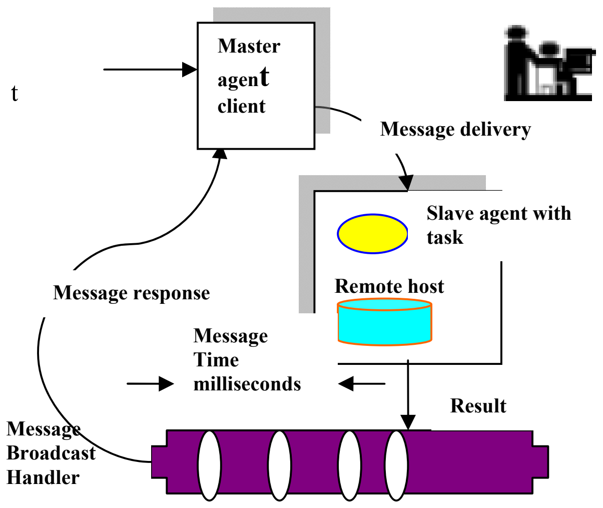


Figure 1. Architecture of Master Slave Task Pattern Message Collaboration

In our system, MVS master slave pattern works with a V-agent in which three types of gents are implemented. They differ mainly in the different roles they can cover and/or they can offer computations. Fig 2, demonstrates MVS design model. Roles of core agents are given as follows:

A. *Master Agents*

These are task handlers in the system. They are created on set of clients. The user uses Java Execution Environment (JEE) platform consists of Tahiti server and Java runtime of Aglet development Kit. After the user instantiates interface, it creates slave agents, reads the database for the list of servers, and check the availability of servers it sends the slave agents to those servers.

B. *V-Agent*

This is our core agent, it instantiates connection between master agents and slave agents, it records message response time from slave agents to delivery master agents messages, it computes total message delivered by each master agent to set of slave agents, and computes total number of received messages by slave agents from set of master agents.

The mediator V-agent acts as a broker to utilizes master and slave agents parameters concerning: message delivery time, message response time, total number of master delivery messages, and total number of slave response messages to build a Message Broadcast Model (MBM) which will be used to optimize and minimize the time between master agents and slave agents, and to better distribute total number of delivery messages by each master agent to slave agents.

C. *Slave Agents*

These are mobile agents which migrates and functioning on servers. They response to master messages, at remote site slave agents receives multiple messages for given tasks and sends results to master agents by embedding it into message handler. Each message registered with the handler, V-agent is responsible to record its time it received by the slave, and the corresponding slave which sends the message and return the result to the master agent.

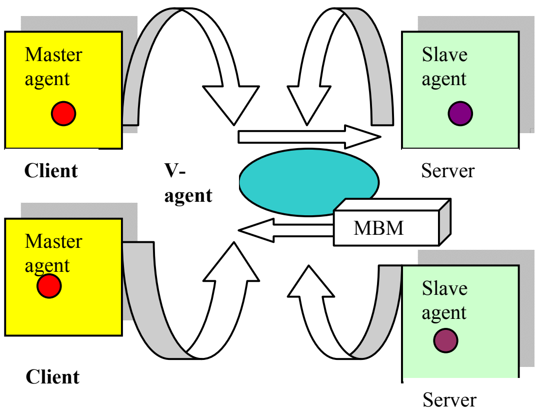


Figure 2. MVS Architectural Design Model

5. Message Broadcast Model (MBM)

To address our work for establishing MBM, consider set of clients C_k, involves set of master agents M_k, where $C_k=\{C_{1m1}, C_{2m2},\dots\dots\dots, C_{kmk}\}$, furthermore, consider a set of servers S_n are distributed across network involves a set of created slave agents, $S_n=\{S_{1s1}, S_{2s2}, \dots\dots\dots, S_{nsn}\}$. Suppose MD_{mi} is the total number of messages to be delivered from the master agent (m_i) to slave agents carrying out some tasks on server side, and suppose MR_{sj} is the total number of response messages that a slave agent (s_j) responds from all master agents in it's broadcast queue.

VAM optimization proposed that total messages that to be delivered by all masters are equal to total response messages in broadcast messaging queue by slave agents, in other words:

$$\sum_{i=1}^{k} MD_{mi} = \sum_{j=1}^{n} MR_{sj}$$

Now, suppose that t_{misj} is the estimated response time of a slave agent (s_j) performing a task on a behalf of a master agent (m_i) and λ_{misj} is number of messages delivered by a master agent (m_i) to a slave agent (s_j).

Based on these assumptions, we suppose the Message Broadcast Model (MBM) that is based on the Message Delivery/Response Matrix (MDRM) between master agents and slave agents as shown in Fig 3.

	S_{1s1}	S_{2s2}	……	……..	S_{nsn}	MD
C_{1m1}	$\mathbf{t_{m1s1}}$ $\boldsymbol{\lambda_{m1s1}}$	$\mathbf{t_{m2s2}}$ $\boldsymbol{\lambda_{m1s2}}$	…….	…….	$\mathbf{t_{m1sn}}$ $\boldsymbol{\lambda_{m1sn}}$	MD_{m1}
C_{2m2}	$\mathbf{t_{m2s1}}$ $\boldsymbol{\lambda_{m2s1}}$	$\mathbf{t_{m2s2}}$ $\boldsymbol{\lambda_{m2s2}}$	…….	…….	$\mathbf{t_{m2sn}}$ $\boldsymbol{\lambda_{m2sn}}$	MD_{m2}
. .	• •	• •	…….	…….	• •	. .
. .	• •	• •	……	…….	• •	. .
C_{kmk}	$\mathbf{t_{mks1}}$ $\boldsymbol{\lambda_{mks1}}$	$\mathbf{t_{mks1}}$ $\boldsymbol{\lambda_{mks1}}$			$\mathbf{t_{mksn}}$ $\boldsymbol{\lambda_{mksn}}$	MD_{mk}
MR	MR_{s1}	MR_{s2}	…….	…….	MR_{sn}	

Figure 3. Message Delivery/Response Matrix (MDRM)

Now, we can formulate the MBM model that follows a Linear Programming model [9] with a set of equations:

- Determine the objective function that should be minimized to achieve optimization.
- Set out model constraints.
- Values of the model should be non negative.

Now, we set out MBM model corresponding to MDRM as follows:

The objective function, constraints, and non negativity condition are given by equation (1), equation (2), equation (3), and equation (4) respectively as follows:

$$\min(t) = \sum_{i=1}^{k}\sum_{j=1}^{n} \lambda_{misj} t_{ij} \quad (1)$$

Subject to (constraints)

$$\sum_{j=1}^{n} \lambda_{misj} = MD_{mi} \quad i = 1,2,\dots..,k \quad (2)$$

$$\sum_{i=1}^{k} \lambda_{misj} = MR_{sj} \quad j = 1,2,\dots\dots.,n \quad (3)$$

and

$$\lambda_{misj} \ge 0 \quad \textit{For all i and j} \quad (4)$$

In the subsequent sections, we will demonstrate how to optimize the objective function using a standard approach in operations research techniques called VAM. This will demonstrate a pervasive role of V-agent coding using VAM. The performance is measured between master agents and slave agents according to V-agent results.

6. MBM Solving Using V-Agent

In our research, mobile agent performance optimization for message distribution across set of slaves, V-agent utilizes VAM which facilitates an optimal solution or good initial solution. VAM is applied on MDRM and is based on the following basic rules.

- Find Penalty Time (PT) between two successive time values in each row and column.
- Choose the larges PT among rows and columns.
- Fill the cell with master message delivery and slave message response.
- Repeat the process until all cells have been filled.
- Calculate the optimal time using equation (1).

The process for obtaining optimal solution using VAM gives an initial solution to MDRM. This measures the performance between master agents and slave agents coordination mechanism as to be given in the subsequent sections with an implementation scenario.

7. Implementation Scenario

To demonstrate how V-agent works, we present a MDRM which utilizes results from master agents to slave agents coordination tasks in message broadcast handler, as given in Fig. 4. Suppose a set of three master agents {m1, m2, m3} with set of total message to be delivered {400, 600, 1000} to set of three slave agents {s1, s2, s3} with total message response {300, 900, 800} respectively. The allocated time for slave message response for one message in milliseconds is summarized in MDRM as below:

	S1	S2	S3	MD
m1	31 λ_{m1s1}	21 λ_{m1s2}	42 λ_{m1s3}	400
m2	20 λ_{m2s1}	21 λ_{m2s2}	30 λ_{m2s3}	1000
m3	23 λ_{mks1}	20 λ_{mks1}	15 λ_{mks3}	600
MR	300	900	800	2000

Figure 4. MDRM Implementation Scenario

Now, we want to solve MDRM to find the optimal solution for the distribution of total master agent's messages to be delivered to slave agents based on slave messages response and for each response message time, number of messages to be assigned from each master agent to slave agents. This can be done by find values of $\{ \lambda_{m1s1}, \lambda_{m1s2}, \ldots\ldots, \lambda_{mks3} \}$ using V-agent.

V-agent starts the process by computing PT among rows and columns leads to choosing the largest PT, if more than one PT values are equal then V-agent would take any PT. Hence the first PT is 15 in the third column. Looking for the smallest allocated time in this column (15). V-agent coordinates the distribution of messages between (m3, s3). So it optimizes the total messages to be delivered by m3, which is 600 and s3 capability to responding is 800. so V-agent will assign the total number of messages to s3 and dispose itself from the queue.

The process is iterative in such case V-agent starts to compute another PT among rows and columns leading to choose the third column with PT is 12. V-agent would choose the smallest allocated time in that column (30), and it optimizes the total messages between (m2, s3). Therefore, V-agent will assign the remaining messages to s3 and s3 will dispose itself from the queue.

V-agent will continue iterates the process until it computes the optimized MDRM given in Fig. 5

	S1	S2	S3	MD
m1	31	21 400	42	400
m2	20 300	21 500	30 200	1000
m3	23	20	15 600	600
MR	300	900	800	2000

Figure 5. Optimized MDRM

The performance optimized time (minimized time) is computed by V-agent according to equation (1) of MBM model as follows:

t= 39900 milliseconds

Analysis of performance optimized time between master agents and slave agents is given in Fig. 6 shows that best time allocated would be when m2 coordinates messages with three salve agents s1,s2, and s3, when the master agent m1 coordinate messages with the second slave agent s2, and the master agent m3 coordination with the third slave s3.

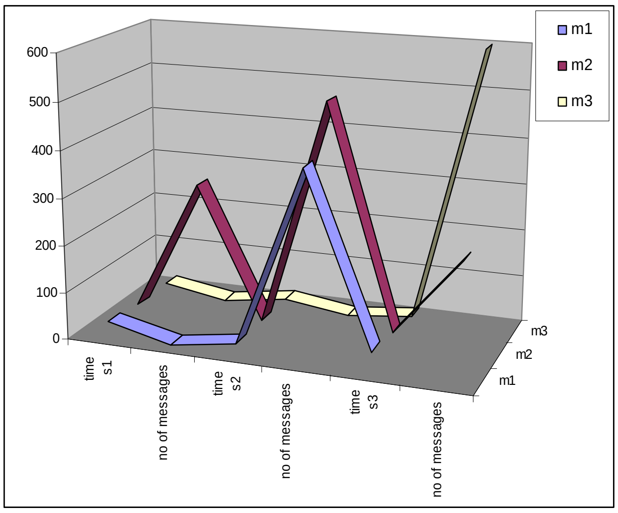
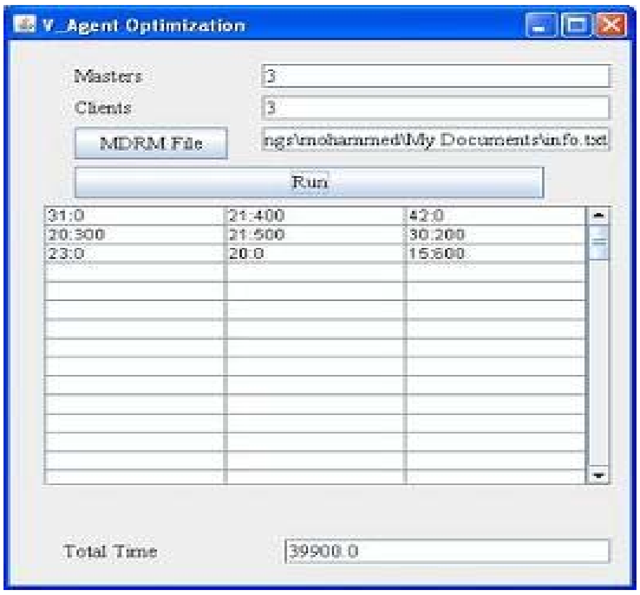


Figure 6. Performance Optimization Analysis between Master and Slave Agents Message Coordination

The analysis scenario discussed in previous section can be given in Fig. 7. A sample screenshot shows V-agent optimization for MDRM .



Masters 3

Clients 3

ngs\mohammed\My Documents\info.txt

31:0	21:400	42:0
20:300	21:500	30:200
23:0	20:0	15:600

Total Time 39900.0

Figure 7. V-Agent Optimization for MDRM

8. Coding V-Agent

In our model, there are mainly three agents working to perform performance optimization: Master agent, Slave agent and V-Agent.

V-Agent is the main agent and it is developed using Java programming language and consists of two main classes (Cell class and VAgent class).

The primary function of V-Agent is to optimize MDRM in addition to the total time needed to send the messages between master agents and slave agents using VAM.

Our agent takes as an input the MDRM(Message Delivery/Response Matrix) which is represented as a matrix of cells, each cell of the matrix is of type Cell(Cell.java) and consists of (Time and Allocation). At the beginning the matrix will be filled in by cells contain the time only.

```
public class Cell {
                  // Attribute of Cell
                 private int time=0;
                 private int allocation=0;
               //Default Constructor
               public Cell(){
              setTime(0);
              allocate(0);
}
          // initialize constructor
public Cell(int time,int allocate){
               setTime(time);
              allocate(allocate);
}
        // set methods
public void setTime(int time){
     this.time=time;
}
public void allocate(int amount){
        allocation+=amount;
}
        // get methods
public int getTime(){
          return time;
}
public int getAllocation(){
          return allocation;
}
             // the value of the cell
public int cellValue(){
          return time*allocation;
     }
}
```

The processing of the matrix using the methods of VAgent class (VAgent.java), the MDRM will contain each cell with its allocated messages according to VAM. optimized MDRM.parts of V-Agent.java is given as follows:

```
 // this method will setup the MDRM
   public  void setupVogel(){
           fillDemand();
           fillSupply();
```

```
        fillTimeMatrix();
    } // this carries out the optimized MDRM
    public void process(){
        Target goal;
         int amount=0;
        while(sumOfDemands>0){
            goal = iterate();

if(demand[goal.getCol()]>=supply[goal.getRow()]){
times[goal.getRow()][goal.getCol()].allocate(supply[goal.
getRow()]);
   demand[goal.getCol()]-=supply[goal.getRow()];
                amount=supply[goal.getRow()];
                supply[goal.getRow()]=0;
            }

else{    times[goal.getRow()][goal.getCol()].allocate(dem
and[goal.getCol()]);

supply[goal.getRow()]=demand[goal.getCol()];
                amount=demand[goal.getCol()];
                demand[goal.getCol()]=0;
            }
            sumOfDemands-=amount;
        }
    }
```

This method represents one iteration of VAM on the matrix and it will return the targeted cell each time

```
  private Target iterate(){
        final int constant = 999;
        int min,next_min;
        int diff;
        int max_diff=-1;
        int c=0,row=0,col=0;
        //iterate through the rows.
        for(int i=0;i<noOfClients;i++){
            //Rows
            if(supply[i]>0){
                min=constant;
                next_min=constant;
            for(int j=0;j<noOfMasters;j++)
                //cols
                if(demand[j]>0){
                    if(times[i][j].getTime()<min){
                        next_min=min;
                        min=times[i][j].getTime();
                        c=j;
                    }
                    else if(times[i][j].getTime()<next_min)
                        next_min=times[i][j].getTime();
                }
            diff = next_min-min;
            if(diff>max_diff){
                max_diff=diff;
                row=i;
                col=c;
            }
          }
        }
        //iterate through the columns
        for(int i=0;i<noOfMasters;i++){
            //cols
            if(demand[i]>0){
                min=constant;
                next_min=constant;

            for(int j=0;j<noOfClients;j++)
                //rows
                if(supply[j]>0){
                    if(times[j][i].getTime()<min){
                        next_min=min;
                        min=times[j][i].getTime();
                        c=j;
                    }
                    else if(times[j][i].getTime()<next_min)
                        next_min=times[j][i].getTime();
                }
            diff = next_min-min;
            if(diff>max_diff){
                max_diff=diff;
                row=c;
                col=i;
            }
          }
        }
        return new Target(row,col);
    }
//this method will return the optimized total time
according to VAM
        public double calculateTime(){
        int time=0;
        for(int i=0;i<noOfClients;i++)
         for(int j=0;j<noOfMasters;j++)
            time+=times[i][j].cellValue();
          return time;
    }
```

9. Conclusions and Future Directions

In this paper, we have developed a new computational model for mobile agents, namely the Message Broadcast Model (MBM). This model is based on a set of master agents working on client side and deliver messages to a set of slave agents carrying out tasks and working on server side. To solve this model, we explore a broker agent, called V-agent that utilizes VAM for time performance optimization between master agents and slave agents message coordination. V-agent uses Message Response/Delivery Response Matrix (MDRM) and carries out an iterative process to do optimization.

Furthermore, analysis of time performance optimization has been reported.

We will comment some future direction trends based on MBM as follows:

- Explore new approaches of time performance optimization using advanced approaches such as stepping stones (sinuous path) and coefficient of multipliers.
- Compare results of different approaches to V-agent approach.
- Build advanced design pattern Optimization Patterns for mobile agents based on MVS architecture and master slave design pattern.
- Develop a generic model that can be generalized in advance with mobile agent design patterns such as coordination patterns, and mobile agent task design patterns.

However, Optimization patterns are the next step towards the communication and improvement that we should developed to support mobility concepts, thus providing more support for the development of distributed information system applications that may benefit from code mobility.

We will extend this work by formalizing optimization mobility patterns. These patterns, and more significantly the skeletons, will be extended to ensure the reliability and fault tolerance of the mobile computations.

10. Acknowledgment

We would like to owe thanks to Applied Science University (ASU), Faculty of Information Technology and the department of Computer Science, Jordan for supporting this research.

References

[1] A.Banerjee, S. Bandyopadyay. "Paradigms for Reliable Communication Protocols in Mobile Agent Based Systems". *In Proceeding 31st Annual Hawaii International Conference on System Science. Vol. 7, 1998.*

[2] D.Chess, C.Harrison, and A. Kershenbaurn. "Mobile Agents: are They a Good Idea?" *IBM Research Division, T.J. Waston Research Center, New York, March,1995,URL: www.cs.dartmouth.edu/?agent/papers/chapter.ps.z*

[3] L. Danny. M Oshima, "Programming and Deploying Java Mobile Agents with Aglets". *Addison Wesley Inc. USA, 1998.*

[4] G. Cabri, L. Ferrari, L. Leonard. "Towards The Use of Mobile Agent Based Systems". *Enabling Technologies Infrastructure for Collaborative Enterprises. 13th IEEE International Workshop, 2004, pages:27-32*

[5] I Sommerville "Software Engineering". *7th edition, Person Education Limited, USA, 2004*

[6] K.B. Mandwake, G.A.Patl. "Performance Analysis of Parallel Server Versus Parallel Mobile Agent Model". *Proceeding of World Academy of Science Engineering and Technology. PWASET Volume 28, 2008, pages: 374-377.*

[7] K. Yasser, A.Hesham, N. Elmahdi, S. Allola, H. Ahmad. "Optimizing Mobile Agents Migration Based on Decision Tree Learnin". Proceedings of World Academy of Science Engineering and Technology. PWASET Volume 22, 2007, pages: 564-570.

[8] P. Braun, W. Rossak."Mobile Agents: Basic Concepts, Mobility Models and the Tracy Toolkit". *Centre for Intelligent & Multi-Agent Systems, Morgan Kaufmann publishers, Australia, 2004.*

[9] S. Frederick, H. Lieberman. "Introduction to Operations Research" *8th edition, McGraw-Hill Education, USA, 2005.*

Biographies

Faiz Al-Shrouf received his Bsc from KSA University in Computer Science, Msc in Mathematical Statistics from Yarmouk University, and ph.D in Software Engineering from University Science of Malaysia. He is currently working as an Assistant Professor at Faculty of IT at Applied Science University/Jordan. His main research interests: Agent Oriented Engineering, Mobile Agents, Software Design Patterns, and M-Commerce.

Mohammed Eshtay received his Bsc and Msc from University of Jordan/Amman in Computer Science, he has 8+ years of experience in Academic and Industrial Fields, and he is currently working as a lecturer in Applied Science University.
He is doing research in Agent Oriented Engineering, Object Oriented Languages, Web semantic, and Component Based Software Development.

Khaled Abu-Humaidan was born in 1976. He received his Bsc degree in Computer Science from Applied Science University / Jordan and Msc degree in computer Science from Free University of Brussels / Belgium. He is currently working as a lecturer at Applied Science University Faculty of IT. His main research interests: Software Engineering, Software testing and validation, Agent Oriented Engineering